VP 講習会テキスト

(MSP 編)

~ VP 上での Fortran プログラムの実行 ~

1996 年 5 月 31 日

九州大学大型計算機センター

MSP/Fortran を VP で実行するには

肥田木 直子 [*]　　渡部 善隆 [†]

目次

[*]九州大学大型計算機センター・システム管理掛
[†]九州大学大型計算機センター・研究開発部

記号の見方

・［　］は括弧で括られた部分が省略可能なことを意味する

・{ a | b | c } は括弧で括られた中のどれかを指定することを意味する

・アンダーラインは指定がない場合の省略値を表す

第 I 部
基礎編

1　はじめに

本稿は大規模な Fortran プログラムを VP で実行するための解説である。大規模な Fortran プログラムとは「配列が大きい」という意味であって、プログラムの長さではない。配列が大きいと、記憶領域(リージョンサイズ)がたくさん必要になる。手持ちのパーソナルコンピュータ、ワークステーションで実行不可能な大規模 Fortran プログラムを実行可能にするため VP は存在している。

プログラム言語を研究・開発する人を除き、計算機を利用して科学技術計算をする人にとっては、OS や使用言語などは何でも構わないはずである (と思う)。大事なのはアルゴリズムであって、Fortran で計算を行なおうが C++ でやろうが Pascal でやろうが、結果が得られれば論文として全く等価である (と思う)。大事なのは計算結果であって、どんな言語でプログラムを組んだかとか、実行時間がどれくらいかかったかなどは「ご苦労様」の一言をもらうだけで何の業績にもならない (場合が多い)。ユーザーが計算機に要求することは、高速な処理と美しいグラフィックである (場合が多い)。

大規模プログラム実行のため、ユーザーが越えなければならないハードルは次の 3 つである。

MSP のエディタの使い方をマスターする
: MSP のデータセット(ファイルのこと)を編集するエディタの知識は無条件で必要。EDIT コマンドで起動されるものと、PFD/EDIT の二つのエディタがある。双方 [9] を読みながら勉強して頂きたい。

本稿を読んでジョブ制御文を書けるようになる
: VP での Fortran プログラムの実行は全てバックグラウンドで行なわれる。そのためには「これこれの処理をお願いします」とシステムに伝える文章をエディタで作成し、依頼する必要がある。本稿の主目的は、このジョブ制御文の書き方にある。

MSP のコマンドを少し知る
: プログラムをバックグラウンドで流して、結果を取り出す過程では幾つかのコマンドを入力する必要がある。MSP を利用する人にとって [9], [12] は必携。

MSP に関しての全くの初心者は必ず [9], [12], [8] を併読されたい。ともに、各連絡所を通じて九州大学大型計算機センター・共同利用掛に申し込むことで入手可能。センターを利用するからには、センター発行の利用の手引は是非手元に置くべきである。

大規模計算の結果は、以上のハードルを越えた後に得られる。気遅れした読者は「でかい計算をするには VP だな」とだけ記憶し、以下を読み飛ばされて結構である。研究者の常識として、〆切前にしか仕事ははかどらないものである。必要に迫られてから読んでも十分に間に合う (と思う)。

2 九州大学大型計算機センターの計算機について

2.1 MSP/Fortran を利用できるシステムの構成

九州大学大型計算機センターで MSP/Fortran [1]を利用できる計算機には、汎用コンピュータ Fujitsu M1800/20U とベクトルプロセッサ (Vector Processor) Fujitsu VP2600/10 （以下 VP と呼ぶ）がある。汎用コンピュータ M1800 上では直接 TSS ジョブを投入する事が可能であり、対話的に Fortran プログラムの処理が可能である。また、汎用コンピュータ上でバッチ処理（バックグラウンドの処理）も可能である。

一方、 VP は M1800 を経由してバッチ形態のジョブしか投入できない。また VP では MT（磁気テープ）やグラフィックスディスプレイ装置は使用できない。

2.2 VP のジョブクラスについて

VP は処理されるジョブの規模に応じて 3 つのクラス (A, B, V) に分けられる。ジョブクラスの指定は、ユーザがジョブ制御文に指定する必要がある (cf.[2] 5.2)。 1995 年 6 月現在の VP のジョブクラスと制限値は下表の通り。

<table>
<tr><th>ジョブクラス</th><th>A</th><th>B</th><th colspan="2">V</th></tr>
<tr><td>CPU 時間 (分)</td><td>10</td><td>180</td><td>10</td><td><u>180</u></td></tr>
<tr><td>ファイルアクセス (万回)</td><td>20</td><td colspan="3">50</td></tr>
<tr><td>リージョンサイズ (MB)</td><td colspan="2">50</td><td colspan="2">300
(SSU 500)</td></tr>
</table>

表中の下線部は、指定のない場合の省略値を表す。また、 SSU(System Storage Unit) については応用編を参照。

2.3 ジョブクラスの設定

ジョブクラスは、バッチジョブを記述するジョブ制御文 (JCL) に指定する大事なパラメータである。 VP での Fortran プログラムの実行では A クラスのジョブが最も早く処理される。理由は、ジョブの規模が B, V クラスより小さいのでセンター側で処理の優先度を高くしているからである。また、 V クラスでも `time` パラメータを 10 分以下に指定することで優先的にジョブが処理される (cf. 5.2)。

ところで、ジョブクラスを指定する時、何を基準に考えたらいいのか？それは最大リージョンサイズである。数値計算は最終的に何らかの線形化演算 (行列を力ずくで処理すること) が必要になる場合が多い。従って VP ジョブのリージョンは、ほとんどが大規模配列で占められる。以下の計算例の様に、どのジョブクラスが適当であるかを各自が判断し指定する。

【計算例】
1000×1000 の倍精度の配列が１個の場合
1000×1000×8 (byte) = 8,000,000（約８ＭＢ）

[1]センターがサポートしている OS の一つである MSP で動作する Fortran システムの総称 (cf.[21])
[2]confer（参照せよ）の意味.

その他の必要な領域を考えても、この場合制限値が 50M の
A か B クラスで適当と思われる。

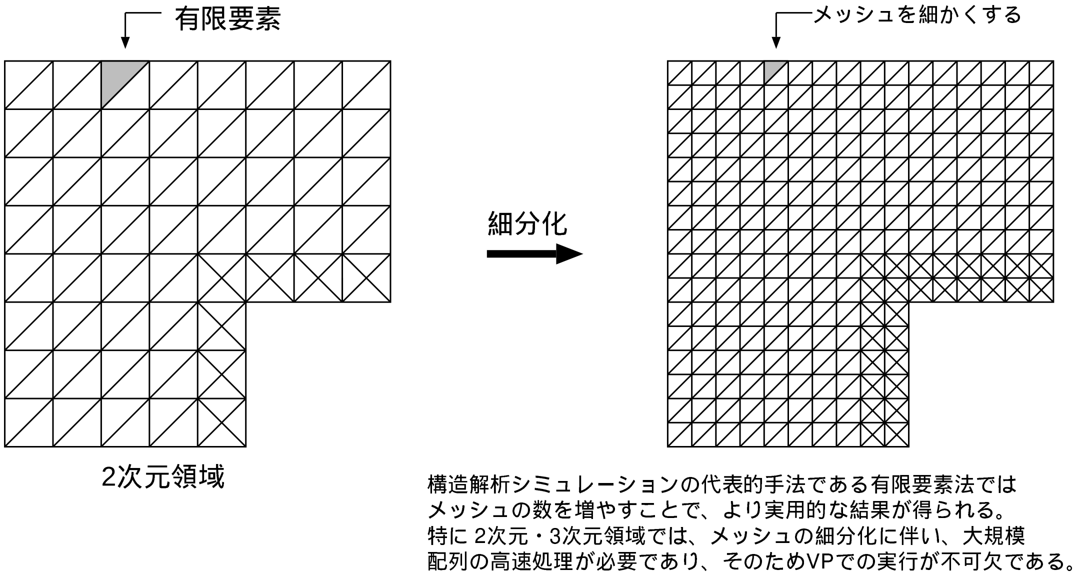


図 1: VP を用いる必要性

2.4 制限値を超えるジョブについて

計算を行ないたい Fortran プログラムが V ジョブの制限値を超えてしまった場合は、 D ジョブ (要審査ジョブ) として提出する方法がある。

方法は、センター 2 階受付横にある専用の用紙に必要事項を記入して受付に提出する。遠隔地の人は共同利用掛を通して依頼する。依頼する時は以下のことに注意。

- VP の要審査ジョブの最大リージョンは 400MB である。
- 依頼のあったジョブはセンターが審査して流すので、制御文を記述したデータセット名と使用システム (VP もしくは汎用機の別) を明記すること。
- 繁忙期 (12 月 ~ 2 月) は使用者が多いため、長く順番待ちの状態になることがある。出来るだけ繁忙期は避けた方が無難である。そのためには論文を秋までに完成すれば良い。
- 実行後の結果は、センター 2 階ロータリーテーブルのある部屋の仕分け棚の「D ジョブ」のところに仕分けておく。出力後のリストについては各自が責任をもって受け取りに来ること。

また、 D ジョブは Fortran プログラム以外の MSP で動作するアプリケーションソフトウェアも処理可能である。例えば、ファイルアクセス回数やリージョンサイズが制限値を越えた MARC, GAUSSIAN, SAS などで更に実行を行ないたい時は、 Fortran と同じ手続きを行なう。

3　VP について

3.1　VP とは

科学技術用プログラムは、配列データを対象とした DO 文の集合になっているものが大変多い。

ベクトル計算機 (VP) とは、こうしたものに対する同一オペレーションの繰り返し処理を、演算パイプラインにより、処理を高速化する機能を持つコンピュータである。従来のスカラー処理は、命令の読み込みから、演算結果を書き込むまでの手順を逐次処理していた。

これに対して VP では、命令の取り出しと結果の書き込みの手順を幾つかに分割して、命令の実行を前の命令の実行とダブってこなしてしまう方式を採用している。この方式は、純粋な並列処理（パラレル処理）とは少し異なり、命令とデータが次々にパイプラインの中を液体のように間をおかずに流れていく様子をとって「パイプライン方式」と呼ばれている。

VP にはこのパイプラインが加減用、掛け算用、割り算用に設けられている。更に、自動的にプログラムをベクトル計算用に書き替える機能を装備したパイプラインもあり、これらのパイプラインが寄ってたかって Fortran プログラムの高速化を目指す。

しかし、この性能を充分に生かすためには、ユーザが各自でプログラムをチューニングする必要がある。また、プログラム自体ベクトル処理が可能な DO 文処理で大部分が占められていなければ、処理は従来のスカラー処理で行なわれることになり、高速化につながらない。

一般に VP は汎用機に比べて高速な処理が実現されるが、ベクトル化率の高い Fortran プログラムを記述し、パイプラインの流れを良くして、 VP の処理をサポートすることも大切である。

3.2　VP 上で動作する Fortran システム

VP 上で動作する Fortran システムには次のソフトウェアがある。

Fortran77 EX/VP
: VP 用オブジェクトモジュールを生成し処理するコンパイラ及びライブラリ全体。

アナライザ
: VP 上における Fortran プログラムの実行状態を解析するツール。 CPU 時間のかかるサブルーチン、 DO ループを解析し、ベクトル化のための情報を提供する。アナライザで解析された情報をもとに、 TSS 形式で TUNER と呼ばれるベクトル化のガイダンスツールを利用することも出来る ([19], [20])。

VP 用サブルーチンパッケージ
: VP 用に開発された科学計算用サブルーチンライブラリ。バッチジョブ処理の段階で自動的に割り当てられるので、ユーザはソースプログラム内からサブルーチンを CALL するだけでよい。富士通株式会社提供の SSL II/VP と、名古屋大学大型計算機センター提供の NUMPAC/VP 等がある。

4　TSS での VP ライブラリの作成

センターの端末やパソコン端末から MSP に LOGON して対話的に処理を行なう TSS 形式では、汎用コンピュータ M1800 が処理を受け持っているため、VP 上での Fortran プログラムの実行は TSS 処理で出来ない。ただし、翻訳と結合・編集を行ない、VP で実行可能なデータセット（ロードモジュール）を作成するまでの手続きは TSS 処理で可能である。

具体的には TSS の `FORT`, `LINK` コマンドによって、VP で実行する為のロードモジュールが作成できる。ただし、作成したロードモジュールを TSS の `RUN` コマンド等で実行出来ないので注意。ここでは、VP 処理に関連したロードモジュール作成の機能のみを説明する。`FORT`, `LINK` コマンドについての詳細は [21], [9], [10], [12] を参照。

4.1　FORT コマンド

TSS から VP 用にコンパイル（翻訳）するためのコマンド。このコマンドは PFD/EDIT からも使用でき、その場合は必須となっているソースデータセット名を省略して行なう。

【入力形式】

FORT　ソースデータセット名
　[VP]
　[OBJ (区分データセット名[3])]
　[AE]
　[NAME]
　[OPT ({ <u>B</u> | E | F })]

【オペランドの説明】

ソースデータセット名	:	ソースプログラムが格納されているデータセット名を指定する。 必須オペランド
VP	:	Fortran77 EX/VP コンパイラを起動させる 省略した場合 Fortran77 EX コンパイラが起動する
OBJ(データセット)	:	指定したデータセットにオブジェクトモジュールを生成する 省略した場合はソースプログラム名から生成されたデータセットに出力
AE	:	拡張リージョンを確保することを指定
NAME	:	ライブラリ用のモジュール作成の際、各副プログラム単位を それぞれ個別のロードモジュールにすることを指定
OPT	:	最適化レベル (cf.12.3) を指定する。省略値は B

[3]MSP の保存形態の一つ。UXP や MS-DOS のディレクトリに対応する。区分データセットは幾つかの「メンバ」に分割される。ロードモジュールでは、サブルーチン名、関数名がメンバ名となるため、区分データセットで作成されなければならない。データセットの概念や作成方法は [9] を参照。

4.2 LINK コマンド

FORT コマンドでコンパイル（翻訳）したもの（オブジェクトモジュール）を結合・編集して、実行用ロードモジュール（カタログドプロシジャ VPGO で実行できるもの）または私用ライブラリを作成するコマンド。

【入力形式】

LINK 入力データセット名
[LOAD (区分データセット名)]
[LIB (区分データセット名)]
[FORTLIB]
[NCAL]

【オペランドの説明】

入力データセット名 : 対象となるオブジェクトモジュール名を指定する。必須オペランド。
LOAD (データセット) : 出力されるロードモジュール名を指定
LIB (データセット) : 結合に必要なライブラリのデータセット名を指定
利用者プログラムの中で、参照しているのに組み込まれていない
副プログラムがある場合、指定されたデータセットから捜す
SSL II/VP, NUMPAC/VP 等を使う場合もここで指定
NUMPAC/VP – 'LIB.NUMVP.LOAD'[4]
SSL II/VP – 'SYS1.SSL2VP'
FORTLIB : Fortran システムライブラリ 'SYS1.FORTLIB'
NCAL : 未解決の外部参照があっても、無視することを指定

【用語の解説】
オブジェクトモジュールとは、 Fortran ソースプログラムを計算機が実行可能な機械語に変換したものである。この変換作業を「翻訳」または「コンパイル」と呼ぶ。 TSS では FORT コマンドによって処理が行なわれる。 Fortran77 EX/VP はこの翻訳作業を行なうプログラムのことであり「コンパイラ」と呼ばれる。

機械語に翻訳した時点では、単にユーザの作ったプログラムを翻訳しただけなので、このままでは実行できない。オブジェクトモジュールに、入出力用の作業や、数学関数、サブルーチンをくっつけて実行可能になる様に補強を行なう。この作業を「結合・編集」または「リンク」と呼ぶ。補強作業は、リンケージエディタ又はローダという名前のプログラムが行なう。TSS では LINK コマンドによって処理が行なわれる。

最終的に出来上がった完全な機械語プログラムを「ロードモジュール」と呼ぶ。

[4]MSP のデータセットの正式な指定方法は、引用符（ ' ）で括って行なう。例えば普段意識せずに EX.FORT がデータセット名だと考えているが、利用者 ID が A79999A の人にとっては'A79999A.EX.FORT' が正式なデータセット名である。

4.3 FORT, LINK コマンドの使用例

【使用例】

- Fortran77 EX/VP コンパイラを使ってソースプログラム VP.FORT の翻訳を行ない、VP.OBJ(MEM1) という名前のオブジェクトモジュールに保存する。

```
 READY
FORT VP.FORT OBJ(VP.OBJ(MEM1)) VP  ↵
```

- ソースプログラム VP.FORT をコンパイルした後、ライブラリとして SSL II/VP を使用[5]して結合編集を行い、VP.LOAD(MEM1) という名前のロードモジュールを作る。

```
 READY
FORT VP.FORT OBJ(VP.OBJ(MEM1)) VP  ↵
LINK VP.OBJ(MEM1) LO(VP.LOAD(MEM1)) LIB('SYS1.SSL2VP') FORTLIB  ↵
```

- サブルーチン SUB.FORT(EX) を私用ライブラリ MYLIB.VP.LOAD に登録する。オブジェクトモジュールは不要なので削除する。

```
 READY
FORT SUB.FORT(EX) OBJ(TMP.OBJ(EX)) VP NAME  ↵
LINK TMP.OBJ(EX) LO(MYLIB.VP.LOAD) NCAL  ↵
DEL TMP.OBJ  ↵
```

  SUB.FORT(EX) に記述された副プログラム名が MYLIB.VP.LOAD のメンバ名として登録される。

【注意事項】

- Fortran77 EX/VP で翻訳、作成したロードモジュールは VP 用のモジュールであり、TSS で実行出来ない。上記例で作成された VP.LOAD(MEM1) を CALL コマンドで実行しようとすると、異常終了する。

- TSS で実行可能なロードモジュールを作成するには、FORT コマンドの VP オプションを指定しないで翻訳を行なうこと。ただし、翻訳は汎用コンピュータ側で (Fortran77 EX) で行なわれ、VP 用のオブジェクトコードは出力されない。汎用機で翻訳した私用ライブラリは VP でも実行可能であるが、ベクトル計算用のライブラリでないため、実行時間が遅くなる可能性がある。

  従って、より高速の実行を望まれる場合は、デバック用の私用サブルーチンを汎用機側で作成し、TSS で十分なデバッグを行なった後に再度 VP 用にプログラムの翻訳を行ない、ベクトル計算向きのライブラリを作成すること。

[5]ソースプログラム内で SSL II/VP のサブルーチンを CALL している場合に指定する。SSL II/VP を CALL していない場合は指定する必要はない。

4.4 私用ライブラリ

4.3 で述べた「私用ライブラリ」とは、文字通り自分用のライブラリとして利用者個人が作成するデータセットのことである。私用ライブラリの中にはサブルーチン、関数が格納されている。データセットは区分編成であり、 Fortran プログラムに記述したサブルーチン、関数名がそのまま区分データセットのメンバ名になる。

```
                                           私用ライブラリ  ~ .LOAD
 SUBROUTINE   SUB1                         メンバ名
 SUBROUTINE   SUB2        翻訳              SUB1      作成した自分用ライブラリの
       :                  ───→             SUB2      メンバ名がサブルーチン / 関数名に対応する
                          編集・結合          :
 FUNCTION   FUNC1                           FUNC1
 FUNCTION   FUNC2                           FUNC2
       :                                      :
```

頻繁に使用するサブルーチン、関数などは、自分用のライブラリに保存する。 VP での実行時にそのライブラリを指定することで、 MAIN プログラムに副プログラムを含めずに CALL するだけでよい。

【注意事項】

- 登録する副プログラムの名前は、そのまま私用ライブラリのメンバ名として登録される。このため、 Fortran77 EX の組み込み関数 (sin, cos, max, etc.) や、 SSL II/VP, NUMPAC/VP のサブルーチン名と重複しないように名前を工夫した方が良い。

Fujitsu VP2600/10

5　バッチ処理での Fortran プログラムの実行

5.1　バッチジョブ

リアルタイムに処理する TSS と違って、バッチジョブは制御文といわれるジョブを処理するための情報を設定するデータセットが必要である。その制御文を投入すると、M1800 を通して VP に処理が依頼され、処理終了後はまた M1800 を通して結果を取り出す。

バッチ処理のメリットは、TSS で処理するより制限値がはるかに大きいことや、ジョブを投入した後でもセッション自体は自由に使用できること、またエラー情報が詳しいこと等があげられる。

5.2　ジョブ制御文入門

バッチジョブについての詳細は [8] を参照。本稿では、Fortran プログラムのバッチ処理という目的だけに的を絞ってジョブ制御文を解説する。

5.2.1　ジョブ制御文とは

バッチジョブを実行させるために計算機に対して、実行するプログラム名や使用するデータセット名、出力するデータセット名など、ジョブ処理の上で必要な情報を与えるのに用いられるのがジョブ制御文である。ジョブ制御文を記述する言語体系をジョブ制御言語（JCL : Job Control Language ）とよぶ。

5.2.2　ジョブ制御文の構成

カタログドプロシジャ FORT を利用したジョブ制御文の構成例を以下に示す。ジョブは JOB 文で始まり、空文で終わる。

下記例は Fortran ソースプログラム `EXAMPLE.FORT` を、入力データ（`READ(5,*)` で読み込むデータ）`EXAMPLE.DATA` を用いて実行する制御文。

```
//A79999A1 JOB CLASS=A                               <--- (1)
//*  TEST JOB FOR FEM                                <--- (2)
//     EXEC FORT,STEP=CLG,OPT=E,VP=YES               <--- (3)
//FORT.SYSIN DD DSN=A79999A.EXAMPLE.FORT,DISP=SHR  <--- (4)
//GO.SYSIN   DD DSN=A79999A.EXAMPLE.DATA,DISP=SHR
//                                                   <--- (5)
```

(1)	JOB 文	–	ジョブの先頭を表す
(2)	注釈文	–	コメント、メモ用
(3)	EXEC 文	–	カタログドプロシジャの呼び出し
(4)	DD 文	–	データセットの割り当て
(5)	空文	–	ジョブの終りを示す

ジョブ制御文は普通 ‘`//`’ から始まる。JOB 文、DD 文はその後にすぐ続けてジョブ名等を記述する。以下順番に機能と設定方法を簡単に述べる。

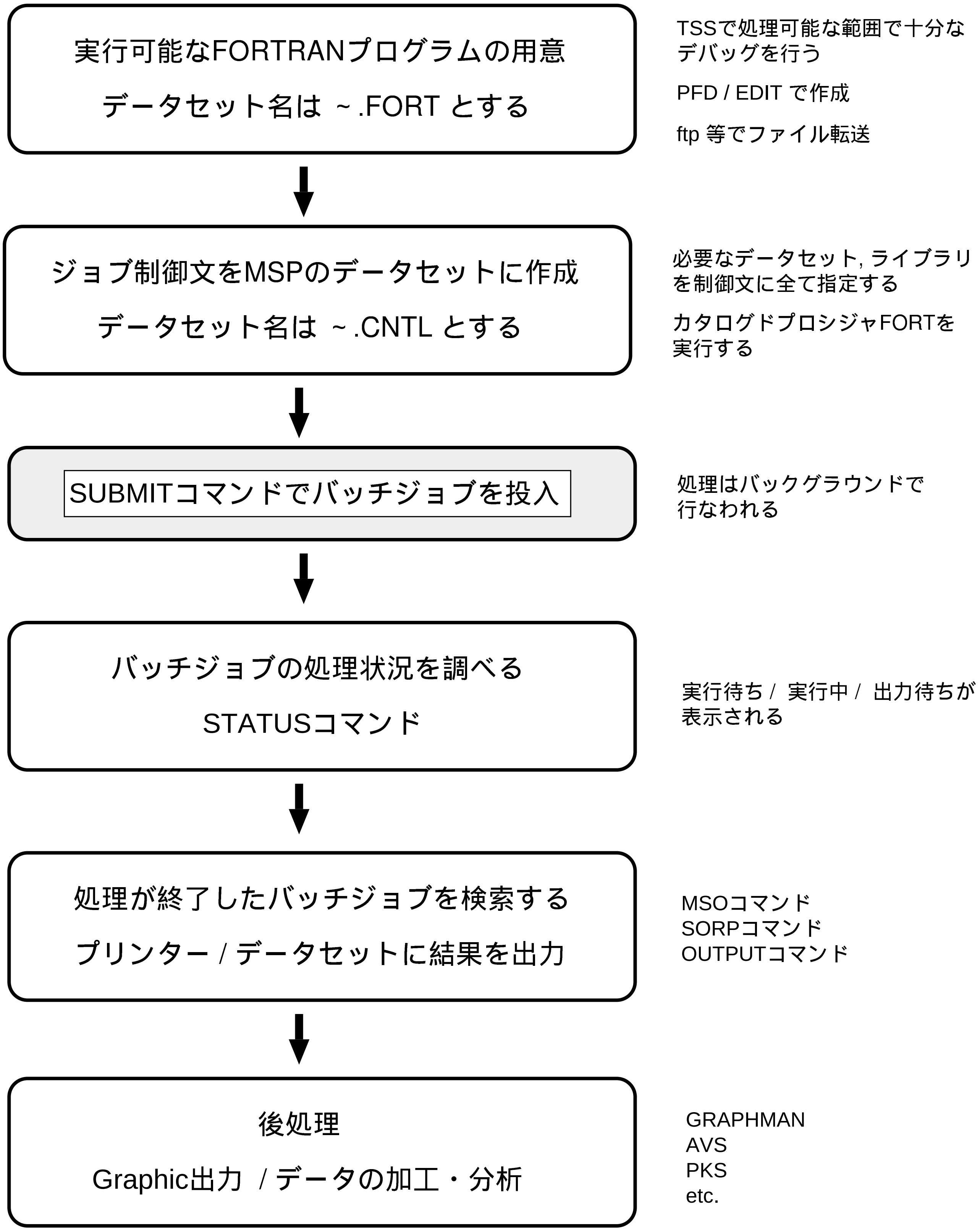

実行可能なFORTRANプログラムの用意
データセット名は ~.FORT とする
TSSで処理可能な範囲で十分なデバッグを行う
PFD / EDIT で作成
ftp 等でファイル転送
ジョブ制御文をMSPのデータセットに作成
データセット名は ~.CNTL とする
必要なデータセット, ライブラリを制御文に全て指定する
カタログドプロシジャFORTを実行する
SUBMITコマンドでバッチジョブを投入
処理はバックグラウンドで行なわれる
バッチジョブの処理状況を調べる
STATUSコマンド
実行待ち / 実行中 / 出力待ちが表示される
処理が終了したバッチジョブを検索する
プリンター / データセットに結果を出力
MSOコマンド
SORPコマンド
OUTPUTコマンド
後処理
Graphic出力 / データの加工・分析
GRAPHMAN
AVS
PKS
etc.

図 2: バッチ処理の流れ

5.2.3 JOB 文

ジョブ文はジョブの先頭を示し、ジョブの属性を与える。 // の後続けてジョブ名を指定する。ジョブ名はセンターから交付された登録番号の後に利用者が任意に識別番号をつける。識別番号は任意の英数字 1 字である。複数のジョブを投入する場合は識別番号を変えて混乱を避けるようにする。

【ジョブ名の例】
A79999A1, A79999AQ など

ジョブ名に続けて必ず JOB と記述する。この後にカンマ (,) で区切りながらオペランドを幾つか指定できる。主なオペランドは次の通り。

【記号パラメータの説明】

CLASS : ジョブクラスを指定する。 VP の場合 A, B, V を指定する。
MSGLEVEL : システムメッセージの出力レベルを指定する。
省略値は MSGLEVEL=(2,0) である。
第 1 サブパラメータはジョブ制御文の出力レベルを指定する。
0 -- JOB 文のみ出力。
1 -- ジョブ制御文とカタログドプロシジャの展開形を出力。
JOB 文でエラーが発見されるとレベル 1 とみなされる。
2 -- ジョブ制御文のみ出力する。
第 2 サブパラメータは入出力装置の割り当ておよびデータセットの後処理に関するメッセージの出力レベルを指定する。
0 -- ジョブステップが異常終了した場合のみ、
入出力の割り当て，後処理に関するメッセージを出力。
1 -- 常にメッセージを出力。
SPARM : 翻訳時メッセージを日本語で出力する。
TIME : ジョブクラスによる制限の範囲内で、ジョブ実行の打ち切り CPU 時間を指定する。省略時は各ジョブクラスの制限値となる。

【使用例】

- MSGLEVEL を (1,1) に指定することで詳細情報を出力させる。デバッグに役立つ

```
//A79999AQ JOB CLASS=V,MSGLEVEL=(1,1)
```

- 翻訳メッセージを日本語で出力させる

```
//A79999AK JOB CLASS=A,SPARM='LANG=J'
```

- CPU 時間を 200 秒 (3 分 20 秒) に設定

```
//A79999AQ JOB CLASS=B,TIME=(3,20)
```

- CPU 時間を 5 分に設定

```
//A79999A1 JOB CLASS=V,TIME=5
```

以上をまとめたジョブ文の指定例を挙げる。ジョブ名は A79999AK とする。詳細情報を出力させ、翻訳メッセージを日本語で出力する。ジョブクラスは V で行ない、ジョブの優先順位を挙げるため CPU 時間を 10 分で区切る。

```
//A79999AK JOB CLASS=V,MSGLEVEL=(1,1),    <--- 継続
//       SPARM='LANG=J',TIME=10
//      EXEC FORT,STEP=CLG,OPT=E,VP=YES
//FORT.SYSIN DD DSN=A79999A.EXAMPLE.FORT,DISP=SHR
//GO.SYSIN   DD DSN=A79999A.EXAMPLE.DATA,DISP=SHR
//
```

1 つの文が数行に渡る時は、‘,’ で行を終了した上で、次の行の第 4 桁から第 16 桁までの任意の行からパラメータの続きを記入する。

5.2.4 EXEC 文

EXEC 文はジョブステップの始まりを示す。呼び出すカタログドプロシジャ（FORT）を指定し、それらを実行させる時の各パラメータを与える。パラメータはカタログドプロシジャ FORT の項を参照。

EXEC 文は // の後に続けてステップ名と呼ばれる 8 文字以内の文字列をユーザが任意に指定できる。ステップ名を省略する場合は空白を数行入れる。ステップ名は普段使用しないので、空白を入れる場合が多い。

5.2.5 DD 文

DD 文はジョブステップで使用するデータセットを定義し、その性質および、割り当てられる入出力装置などを指定する (DD は dataset definition の意味).

// の後に指定される FORT.SYSIN, GO.SYSIN などは DD 名と呼ばれ、 DD 文を識別する名前であり、 プログラムによって決められている。この名前によって、プログラムでアクセスしたいデータセットと実際のデータセットが結び付けられる。一般に省略できない。

例）Fortran ソースプログラムを翻訳，編集結合，実行させる STEP=CLG の場合、
ソースプログラム入力の DD 名は FORT.SYSIN を指定することに決められている。
同じく実行時の入力データの DD 名も GO.SYSIN と決められている。

```
//A79999AQ JOB CLASS=B
//      EXEC FORT,STEP=CLG,OPT=E
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//GO.SYSIN   DD DSN=A79999A.TEST.DATA,DISP=SHR
//
```

DD 名に続く ‘DD’ は、 DD 文であることの宣言である。それに続いてオペランドを指定する。ここでは Fortran の 実行に関する主なオペランドを解説する。

【記号パラメータの説明】

DSN　:　データセットの名前を指定する。

UNIT　:　割り当てるべき入出力装置を指定する。

- PUB　-　保存データセット用直接アクセス記憶装置
- WORK -　一時データセット用直接アクセス記憶装置
- SSU　-　システム記憶装置
- MSD　-　大容量記憶装置（マスストレージ利用申請が必要）

DISP　:　データセットの状態および後処理に関する情報を指定する。

第 1 サブパラメータはデータセットの状態を示す。

- NEW - このジョブステップで新しく作成されることを示す。
- OLD - データセットがすでに存在していることを示す。
  データセットへの書き込みを行う時に指定する。
- MOD - 既存の順データセットに対して、最終データの後に追加して書き込みを行うことを示す。
- SHR - データセットがすでに存在していることを示す。
  データセットの読み込みを行う時に指定する。

第 2 サブパラメータはデータセットの後処理を指定する。

- DELETE - ジョブステップ終了後データセットを消去することを指定。
- KEEP　- データセットを保存することを指定する。
  （磁気テープ上のデータセットに対してのみ）
- PASS　- 後続ジョブステップにデータセットを引き渡すことを示す。
  PASS 指定されたデータセットを受け取れるのは 1 回限り.
- CATLG　- データセットを保存し、かつカタログすることを示す。
  直接アクセス記憶装置上の保存データセットは必ずカタログしなければならない。

SPACE　:　直接アクセス記憶装置上に要求するスペースを指定する。

{ TRK | CYL | 平均ブロック長 }　:　割り当て単位を示す。

TRK を指定する場合 1 トラック当たりの記憶容量は約 47 キロバイトである。

CYL を指定する場合は、1 シリンダ当たり 15TRK である。

初期量　:　新しく作成するデータセットにスペースを割り当てる数を指定する。

増分量（省略値 =0）:

データセットの作成中あるいは拡張中にスペースが不足した場合、
1 回の増分割り当てで拡張すべきトラック数またはブロック数を指定。
増分割り当ては最大 15 回。

ディレクトリブロック数　:

区分データセットを作成するとき、ディレクトリ用のスペースとして、
256 バイトのブロックをいくつ作成するかを指定する。
0 を指定すると順データセットが作成される。

RLSE　:　データセットを CLOSE する時、未使用領域を解放することを指示する。

DCB :　実行時にデータ制御ブロック (DCB) を完成させるために指定する。
DCB サブパラメータ（ブロックサイズ、レコードサイズなどを指定）は、
プログラムにより指定可能なものが定められる。
よく用いられるサブパラメータに LRECL, BLKSIZE, RECFM がある (cf. [9], [8]).

【使用例】

- DD 文で最も使われる例は、 WRITE(6,*), PRINT の出力先をデータセットに指定する場合である。テキスト形式で出力させ、後で出力データをパソコンやワークステーションへ転送することを考えれば、次の様にデータセットを作成し、出力させるのがよい。

  Fortran の実行 STEP は省略値の STEP=CLG と仮定している。

```
//GO.FT06F001 DD DSN=A79999A.RESULT.DATA,UNIT=PUB,
//     DISP=(NEW,CATLG),SPACE=(TRK,(20,10)),
//     DCB=(LRECL=80,BLKSIZE=23440,RECFM=FB)
```

  上の DD 文で GO.FT06F001 が WRITE(6,*) に対応する。出力データセットは 'A79999A.RESULT.DATA' としているが、各自の課題に合わせて修正すること。 UNIT=PUB はデータセットとして保存することを意味する。出力が多すぎて SPACE パラメータで指定した容量を越える場合は SPACE パラメータの (20,10) の値を大きめに修正すること。

  また、注意として'A79999A.RESULT.DATA' は新規作成するので、既存の場合は「既にあるよ」というメッセージを発してエラーとなる (この辺りのファイル管理が MSP の信頼できる所であり、融通のきかないところである)。既存のデータセットに上書きする場合は次のように修正する。

```
//GO.FT06F001 DD DSN=A79999A.RESULT.DATA,DISP=OLD
```

  既存のデータセットに追加書きする時は次のように指定する。

```
//GO.FT06F001 DD DSN=A79999A.RESULT.DATA,DISP=MOD
```

5.2.6 コメント文

'//*' で始まる文をコメント文と呼び、'//*' 以下の記述は無視される。コメント文は、ジョブ制御文として単独に注釈を記入したい時に使用する。

```
//A79999AQ JOB CLASS=B
//     EXEC FORT,STEP=CLG,OPT=E
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//GO.SYSIN   DD DSN=A79999A.TEST.DATA,DISP=SHR
//*GO.SYSIN   DD DSN=A79999A.EX.DATA,DISP=SHR    <-- コメントとして
//                                                   実行されない
```

5.2.7 空文

'//' のみの文を空文と呼ぶ。空文はジョブの終わりを示す。

5.3 カタログドプロシジャ FORT について

Fortran77 EX/VP コンパイラによる翻訳、結合編集、実行を行うのがカタログドプロシジャ (Cataloged Procedure) FORT であり、ジョブ制御文の EXEC 文で呼び出される。このカタログドプロシジャ FORT は、指定によって Fortran77 EX コンパイラによる処理も行う。また、 TSS コマンドの FORT とは全くの別物である。

5.3.1 FORT の入力形式

	主なパラメータ
FORT	[, SYSOUT={ <u>A</u> | K | S | O | U | H }]
	[, STEP={ C | CGO | CG | CL | <u>CLG</u> }]
	[, OPT={ <u>B</u> | E | F }]
	[, PRVLIB='データセット名']
	[, OPTION='コンパイラオプション']
	[, VP=YES]
	[, VREGION= リージョンサイズ]

【記号パラメータの説明】

SYSOUT : 出力クラスを指定する。省略値は A。
STEP : 処理過程を選択する。省略値は CLG。
C ... 翻訳のみ行う。
CGO ... 翻訳、結合編集、実行を 1 ステップで行う。
CG ... 翻訳、ローダによる結合編集、実行を行う。
CL ... 翻訳、リンケージエディタによる結合編集まで行い、私用ライブラリあるいは実行用のロードモジュールを作成する。
CLG ... 翻訳、リンケージエディタによる結合編集、実行を行う。
OPT : 最適化のレベルを指定する。省略値は B。
PRVLIB : 組み込みたい私用ライブラリのデータセット名を指定する。
OPTION : コンパイラオプションのリストを記述する。
VP=YES : VP で実行する時指定する。省略すると M1800 で実行される。
VREGION : VP 実行時のリージョンサイズを MB 単位で指定する。

5.3.2 STEP 値と最適化オプション

センターでは、サービス終了時間になるとジョブ処理を中断し、計算機の電源を切断している。この時、 STEP=CGO, STEP=CG の様に 1 つのステップで複数の処理をおこなうマルチタスクは、実行中断されたジョブは翌日にジョブステップの最初から実行を再開し、前日のジョブステップの CPU 時間が無駄なる。 CPU 時間の長いジョブを実行する場合、必ず省略値である <u>STEP=CLG</u> で実行すること。

VP2600 上で Fortran プログラムを OPT=E, OPT=F で実行する場合は、 OPTION='OPTMSG' を指定することでメッセージが確認できる。大規模なプログラムを実行させる場合、最適化のレベルやプログラムによっては翻訳・実行時間にかなりの差が出る。センターの最適化レベルの省略値は VP2600 は OPT(B) である。特に VP 上での Fortran プログラムの実行では、 [21]

に述べられている副作用の発生する可能性を考慮した上で、OPT(E), OPT(F) を指定することで、実行の高速化が可能である。ユーザ自身で最適化レベルを積極的に設定されたい。

オプション	ジョブステップ	関連する DD 名	データセット
STEP=C	FORT	FORT.SYSIN	ソースプログラム
STEP=CL	FORT LKED	FORT.SYSIN LKED.SYSIN	ソースプログラム リンケージエディタ制御文
STEP=CLG	FORT LKED GO	FORT.SYSIN LKED.SYSIN LKED.SYSLMOD GO.SYSIN	ソースプログラム リンケージエディタ制御文 私用ライブラリあるいは ロードモジュール保存 実行時の入力データ
STEP=CG	FORT LOADGO	FORT.SYSIN LOADGO.SYSIN	ソースプログラム 実行時の入力データ
STEP=CGO	FORTCGO	FORTCGO.SYSIN FORTCGO.SYSGO	ソースプログラム 実行時の入力データ

STEP 値に関する DD 名

5.3.3 制御文の例

制御文は必ず‘*.CNTL’形式の名前でデータセットに作成すること。TSS コマンドとして打ち込むことは出来ない。

- ソースプログラム名 VP.FORT。入力データの入っているデータセット名 VP.DATA。

```
//A79999A1 JOB CLASS=V
//  EXEC FORT,VP=YES,STEP=CLG,OPT=E
//FORT.SYSIN  DD  DSN=A79999A.VP.FORT,DISP=SHR
//GO.SYSIN    DD  DSN=A79999A.VP.DATA,DISP=SHR
//
```

- VP.LOAD(EX) というロードモジュールを新規に作る。CPU 時間は 10 分を指定。ソースプログラム名 VP.FORT(EX)。

```
//A79999A2 JOB CLASS=B,TIME=10
//  EXEC FORT,VP=YES,STEP=CL,OPT=B
//FORT.SYSIN    DD DSN=A79999A.VP.FORT(EX),DISP=SHR
//LKED.SYSLMOD DD DSN=A79999A.VP.LOAD(EX),UNIT=PUB,
//        DISP=(NEW,CATLG),SPACE=(TRK,(10,10,1),RLSE)
//
```

- 幾つかのサブルーチンを記述したデータセット SUB.FORT を翻訳し、VP 用のライブラリとして MYLIB.LOAD という名前の私用ライブラリを新規に作成し、登録する。CPU 時間はそれほどかからないので、クラス A で処理を行なう。

```
//A79999A3 JOB CLASS=A
//  EXEC FORT,VP=YES,STEP=CL,OPTION=NAME,PARM.LKED=NCAL,OPT=B
//FORT.SYSIN  DD DSN=A79999A.SUB.FORT,DISP=SHR
//LKED.SYSLMOD  DD  DSN=A79999A.MYLIB.LOAD,UNIT=PUB,
//       DISP=(NEW,CATLG),SPACE=(TRK,(10,10,5),RLSE)
//
```

- 上で作成した私用ライブラリ MYLIB.FORT を使って Fortran プログラム VP.FORT を実行させる。読み込み用のデータは VP.DATA。

```
//A79999A4 JOB CLASS=A
//  EXEC FORT,VP=YES,STEP=CLG,PRVLIB='A79999A.MYLIB.LOAD',OPT=E
//FORT.SYSIN DD DSN=A79999A.VP.FORT,DISP=SHR
//GO.SYSIN   DD DSN=A79999A.VP.DATA,DISP=SHR
//
```

- ソースプログラム VP.FORT を実行する。出力結果（WRITE(6,*) の出力先）は RESULT.DATA という新しいデータセットに落とす。入力データは VP.DATA から読み込む。

```
//A79999A5 JOB CLASS=V,TIME=180
//  EXEC FORT,VP=YES,STEP=CLG,OPT=B
//FORT.SYSIN  DD DSN=A79999A.VP.FORT,DISP=SHR
//GO.SYSIN    DD DSN=A79999A.VP.DATA,DISP=SHR
//GO.FT06F001 DD DSN=A79999A.RESULT.DATA,UNIT=PUB,
//    DISP=(NEW,CATLG),SPACE=(TRK,(10,10)),
//    DCB=(LRECL=80,BLKSIZE=23440,RECFM=FB)
//
```

RESULT.DATA が既に存在する場合は、次の指定でよい。

```
//GO.FT06F001 DD DSN=A79999A.RESULT.DATA,DISP=SHR
```

データセットが存在するのに DISP=NEW を指定したり、データセットが存在しないのに DISP=SHR を指定したりするとエラーとなる。 WRITE(6,*) の出力先は GO.FT06F001 で特に指定しない場合は、ジョブの実行結果に出力される。ただし、 OUTPUT コマンドを用いることでジョブの実行結果はデータセットへ出力可能なので、そのデータセットをエディタで編集すれば WRITE(6,*) の出力ファイルが作成できる。

5.4 カタログドプロシジャ VPGO

Fortran77 EX/VP コンパイラを使って翻訳、結合編集を行い、作成されたロードモジュールを実行させるのが、カタログドプロシジャ VPGO である。 TSS で作成した VP 用のロードモジュールは、このカタログドプロシジャで実行する。

	パラメータ
VPGO	, PROG= プログラム名 , LOADDS=' 区分データセット名' [, SYSOUT={ A | K | S | O | U | H }] [, VREGION= リージョンサイズ]

記号パラメータの説明

PROG	:	実行するプログラム名を指定する。
LOADDS	:	ロードモジュールが格納されているデータセット名を指定する。
SYSOUT	:	出力クラスを指定する。省略値は A。
VREGION	:	VP 実行時のリージョンサイズを MB 単位で指定する。 省略時は各ジョブクラスの制限値がとられる (cf.2.2)。

【使用例】

- ロードモジュール VP.LOAD のメンバ EX を実行させる。入力データは VP.DATA。 TIME パラメータを 10 分に指定し、処理の優先度をあげる。

```
//A79999A6 JOB CLASS=V,TIME=10
//  EXEC VPGO,PROG=EX,LOADDS='A79999A.VP.LOAD'
//GO.SYSGO DD DSN=A79999A.VP.DATA,DISP=SHR
//
```

【注意事項】

- ロードモジュール名は完全データセットで指定すること。

  × LOADDS=VP.LOAD
  × LOADDS=A79999A.VP.LOAD
  ○ LOADDS='A79999A.VP.LOAD'

- カタログドプロシジャ VPGO は、 VP 用に作成されたロードモジュールを VP で実行するためのものである。汎用機でのロードモジュール実行はカタログドプロシジャ GO で行なう ([8]) こと。

5.5　バッチジョブ関連で用いる TSS コマンド

5.5.1　バッチジョブの依頼 : SUBMIT コマンド

データセットに作成したジョブ制御文は、TSS の SUBMIT コマンドでジョブを依頼する。SUBMIT コマンドはバッチジョブを依頼する唯一のコマンドであり、これを忘れると何も出来ない。最も重要なコマンドである。

【使用例】

- ジョブ制御文 VP.CNTL を依頼する。VP.CNTL には TSS のエディタで作成した '//' で始まるジョブ制御文が記述されている。READY モードから入力する。

```
 READY
SUB VP  ↵          <--- ジョブの依頼
```

  SUBMIT は SUB と省略可能。また VP.CNTL も VP だけで OK である。

- ジョブ制御文は、区分編成でも良い。VP.EX.CNTL という区分データセットのメンバ EX1 に記述されたジョブを依頼する。

```
 READY
SUB VP.EX(EX1)  ↵      <--- CNTL は省略可能
```

- PFD/EDIT 内から直接バッチジョブを依頼する。「SUB」と打ち込むだけで編集中のジョブ制御文が読み込まれ、とても便利である。

```
EDIT --- A79999A.VP.CNTL -------------------------------- COLUMNS 001 072
COMMAND ===> SUB  ↵                                       SCROLL ===> HALF
****** *************************** TOP OF DATA *************V10L30********
000100 //A79999AA JOB CLASS=V
000200 //  EXEC FORT,VP=YES,STEP=CLG,OPT=E
000300 //FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
000400 //
****** ************************** BOTTOM OF DATA *************************
```

```
KEQ56208I ***           A79999AA        : (RECEIVED) ***
      *** A79999AA (J0998) A79999A : (JOB ACCEPTED) *** FIB CN(01)
KEQ56250I JOB A79999AA(JOB00998) SUBMITTED
***
```

5.5.2 STATUS

STATUS コマンドは依頼したバッチジョブの処理状況 (実行待ち、実行中、出力待ち) を表示する。 ST と省略可能である。

```
 READY
ST  ↵
 KEQ56192I JOB A79999A2(JOB06492) IS WAITING FOR EXECUTION      ← 実行待ち
 KEQ56192I JOB A79999A3(JOB06495) IS EXECUTING                  ← 実行中
 KEQ56221I JOB A79999A#(TSU07684) IS EXECUTING ON THIS TERMINAL ← 現 session
 KEQ56192I JOB A79999A1(JOB06490) IS WAITING FOR OUTPUT         ← 出力待ち
```

5.5.3 STATE

現在の日時、現在の TSS ユーザ数、バッチジョブの実行状況および実行待ち件数を表示する。 STATUS コマンドよりもより詳しい処理状況がわかる。

```
 READY
STATE  ↵

TIME=10.04.27  DATE=93.04.05
TSS USER 0011
JOBNAME STEP     SNO CLS REGION E-REGION CPU TIME START    ACCEPT
A79999A GAUSSIAN  1  F  7232KB     93MB 00:05:49 09:51:59 09:51:47  04/15/93

 ***  YOUR JOB WAITING FOR EXECUTION *** AT 04/15/93 10:17;58
 JOBNAME(JOBNO)   CLASS   SYSTEM   ORDER   CLASSTOTAL
A79999AQ(JOB4455)   F      MSP        2       10
A79999AV(JOB3433)   B      VP         6        7

    * INFORMATION OF WAITING JOB EXECUITION * 10:04 ON 04/15/93 ****
    * : JOBCLASS :   A :   B :   C :   F :   V :   G :   L :   N : *
    * +----------+-----+-----+-----+-----+-----+-----+-----+-----+ *
    * :  M1800   :   1 :   2 :   0 :  10 : --- :   0 :   0 :   0 : *
    * :  VP2600  :   2 :   7 : --- : --- :   7 : --- : --- : --- : *
    ****************************************************************
```

5.5.4 MSO, SORP

MSO コマンドは、バッチジョブをメニュー形式で出力検索する。フルスクリーン端末でバッチジョブの操作 (状態表示・検索・消去・出力) がメニュー形式で行える。動作は TTY 端末、FNVT 上でも可能である。

起動は TSS コマンドの MSO で行なう。

```
 ---------------------------< M S O  V02/L02 >--------------- BY KYUSHU UNIV.
                                                                    93.05.27
                      YOUR JOBS ARE AS FOLLOWS.
  (NO.) (JOBNAME)       (STATUS) (RANK) (NO.) (JOBNAME)       (STATUS) (RANK)
      1  A79999A1 (J3362) OUT         ← (1)
    V 2  A79999AF (J3881) EXEC        ← (2)
    V 3  A79999AH (J3883) WAIT    2/3 ← (3)
    M 4  A79999AH (J3882) WAIT    1/5


    SEQ-NO.   ==>  (4)                      SORP-DSPRINT(PF5) DATASET
    OPTION    ==>  (5)   < B,NB,E,NE,C >          ==>   (7)
    NEW CLASS ==>  (6)   < H,O,K,S,E,U OPR-NO >
    TSS       ==>
   ---------------------------------------------------------------------------
             PF1 ==> HELP  PF2 ==> RE-FRESH  PF3 ==> END  ← (8)
```

表示後の画面の説明

(1) 出力待ちのジョブであることを示す。
(2) 実行中のジョブであることを示す。 M は汎用機、 V は VP を意味する。
(3) 実行待ちのジョブであることを示す。分数は待ちの順番を表す。
(4) 画面に表示されているジョブ一覧の番号を選択する。特に 99 を指定すると、
次項 (5) の C オプション、あるいは、 (6) の項を指定した場合に限り、
すべてのジョブが対象となる。ただし 1 度選択したジョブは対象とならない。
(5) 出力検索時の環境を指定する。

空白	.....	SORP コマンドによる出力検索を行う。
B	.....	PFD の BROWSE オプションの出力検索を行う。
NB	.....	PFD の日本語 BROWSE オプションの出力検索を行う.
E	.....	PFD の EDIT オプションの出力検索を行う。
NE	.....	PFD の日本語 EDIT オプション風の出力検索を行う。
C	.....	(4) で選択したジョブをキャンセルする。

(6) ジョブを出力する場合の出力クラス、あるいは、 OPR のプリンタ名を選定する。
(7) SORP コマンド実行時にページ抽出するデータセットを指定する。
(8) PF1 を押すと HELP 画面が表示される。 PF2 を押すとジョブの状態の再検索
および画面の再出力を行う。 PF3 を押すと MSO コマンドは終了する。

【留意事項】

1. TTY 型端末で TSSPFD エミュレータを使用される方は最新のエミュレータを使用すること. 最新のエミュレータはセンター 2 階オープン室にあるので各自でコピーされたい
2. SORP コマンドの使用法については、TSS で MANUAL コマンド（MANUAL SORP）を入力し「SORP コマンド使用の手引」を出力して参照されたい。
3. バッチの出力結果は 2 週間を過ぎると強制的に消去されるので、必要なものはその期限内に出力すること。
4. PFD 3.8 でも検索が可能である。
5. EDIT 風の検索オプションで検索内容を編集しても、実行結果が書き変わることはない。

5.5.5 PFD の OUTLIST 機能を利用したジョブの出力

フルスクリーン端末からメニュー形式でバッチジョブの出力検索を行う。PFD のメニューで選択可能であるが、いきなり READY モードで PFD 3.8 と入力すればジョブが検索できる。

【使用例】

```
 REDAY
PFD 3.8  ↵
```

```
-----------------------<  OUTLIST ユティリティ  >-------------------------
オプション ===>

   D     - ジョブの出力結果をスプールから削除する。
   L     - ジョブの状態を通知する。
   P     - ジョブの出力結果をプリントしてスプールから削除する。
   R     - ジョブの出力クラスを変更する。
   E     - 日本語データを含まないジョブの出力結果を編集する。
   J     - 日本語データを含むジョブの出力結果を編集する。
   N     - 日本語データを含むジョブの出力結果を表示する。
   空白 - 日本語データを含まないジョブの出力結果を表示する。

以下のパラメタを指定して下さい。
  ジョブ名         ===> A79999AA
  SYSOUT クラス    ===>
  ジョブ識別番号    ===> J6492
  新 SYSOUT クラス ===>              （オプション'R' を選択した場合）

  印刷制御文字      ===>              (A-ANSI,M-MACHINE,BLANK-NONE)
-------------------------------------------------------------------------
```

5.5.6 CANCEL

依頼したバッチジョブを取り消す。また、マルチセッション下での TSS セッションのキャンセルも行なえる。 PURGE オペランドを指定すれば、出力待ちのジョブもキャンセル可能。

【使用例】

- ジョブ A79999AB を取り消す。

```
 READY
CANCEL A79999AB  ↵
```

- ジョブ ID が同じ場合は（JOB*****）まで指定する必要がある。
  ジョブ A79999AK(JOB00002)、 A79999AK(JOB00010) のうち、 A79999AK(JOB00002) をキャンセルする。

```
 READY
CANCEL A79999AK(JOB00002)  ↵
```

【注意事項】

- 依頼したジョブの状態を知るにためには、 STATUS コマンド（STATUS あるいは ST）を用いる。

5.5.7 OUTPUT

ジョブの検索機能、キャンセル機能もあるが、主にジョブ出力のデータセットへの保存に用いる。詳細は [12] を参照。省略形 OUT でも使用できる。

【使用例】

- ジョブ A79999AB をデータセット RESULT.OUTLIST に出力する。

```
 READY
OUTPUT A79999AB PRINT(RESULT)  ↵
```

  RESULT.OUTLIST は既存でなくてよい。上記例で A79999AB はデータセットに結果を出力した後、消去される。

  また、.OUTLIST は自動的に付けられる。 PRINT(RESULT.OUTLIST) と指定すると、RESULT.OUTLIST.OUTLIST となるので注意。 .OUTLIST をつけないためには完全データセット名（ex.'A79999AB.RESULT.DATA'）で指定する。

6 VP での実行例

本節では、 VP で Fortran プログラムをバッチ処理する行程を、具体的な例題に沿って述べる。 PFD/EDIT を利用してデータセットの編集を行ない、端末はセンターの端末 (VDS, NDS 端末) 環境に準じた full screen 環境が実現されていることを想定している。利用者の課題番号は `A79999A` とする。

6.1 Step 1 : Fortran プログラムの作成

先ず、実行可能な Fortran プログラムを用意する。例として、 SSL II を用いた以下のサンプルプログラムをあげる。 VP での大規模計算の前提として、 Fortran プログラムの十分なデバッグが必要である。 MSP の TSS 環境は、デバッグに適した環境であるため PFD の EDIT オプションによって、 Fortran プログラムの作成、編集を行なう。また、ある程度の規模のジョブ (50MB 以下) ならば、 TSS で会話的にプログラムの処理が出来る。

```
EDIT --- A79999A.TEST.FORT ---------------------------- COLUMNS 001 072
COMMAND ===> R ↵                                        SCROLL ===> HALF
****** ************************ TOP OF DATA ************V10L30********
000100       PROGRAM MAIN
000200       PARAMETER(N=100)
000210       REAL*8 A(N,N),X(N),Y(N),S/0.0D0/
000300       DO 10 I=1,N
000400       DO 20 J=1,N
000410       A(I,J)=1.0D0/DBLE(I+J-1)  ! HILBERT MATRIX
000420 20    CONTINUE
000421       X(I)=DBLE(I-1)
000430 10    CONTINUE
000600       CALL DMAV(A,N,N,N,X,Y,ICON)
000900       DO 30 I=1,N
001000 30    S=S+Y(I)
001300       WRITE(6,100)S
001400 100   FORMAT(1X,'SOLUTION =',F15.8)
001500       END
****** *********************** BOTTOM OF DATA ************************
```

上記プログラムを PFD 内から `RUN` サブコマンドで実行すると、次のメッセージが端末に表示され、正常にプログラムが動作することが確認される。

```
 Fortran77 EX COMPILER ENTERED
 END OF COMPILATION,HIGHEST SEVERITY CODE=00
 SOLUTION =  4930.93465695
 END OF GO,HIGHEST SEVERITY CODE=00
***
```

小規模な領域で翻訳・実行がうまく行くことを確認した上で、大容量のジョブを実行すると効率が良い。

6.2 Step 2：私用サブルーチンライブラリの作成

Step1 の応用として、自分用のライブラリを作成して、メインプログラムから CALL する場合を挙げる。特に大規模・複雑な計算になると、頻繁に使われるサブルーチン・関数などはライブラリとして保存しておくと、いちいちメインプログラムに書き込む必要がなくなり、すっきりする。

例題では、Step 1 で挙げた例題 `TEST.FORT` を `MAIN.FORT` と、`TEST.FUNC.FORT(EX1)`, `TEST.FUNC.FORT(EX2)` に分割する例を述べる。`TEST.FUNC.FORT` は区分データセットとする。

```
EDIT --- A79999A.MAIN.FORT ---------------------------- COLUMNS 001 072
COMMAND ===>                                            SCROLL ===> HALF
****** ************************ TOP OF DATA ************V10L30********
000100       PROGRAM MAIN
000200       PARAMETER(N=100)
000300       REAL*8 A(N,N),X(N),Y(N),S/0.0D0/
000400       CALL MAKEM(A,N)  ! CALL PRIVATE LIB
000500       DO 10 I=1,N
000600       X(I)=DBLE(I-1)
000700 10    CONTINUE
000800       CALL DMAV(A,N,N,N,X,Y,ICON)
000900       CALL CSUM(Y,S,N) ! CALL PRIVATE LIB
001000       WRITE(6,100)S
001100 100   FORMAT(1X,'SOLUTION =',F15.8)
001200       END
****** *********************** BOTTOM OF DATA ***********************
```

```
EDIT --- A79999A.TEST.FUNC.FORT(EX1) ------------------ COLUMNS 001 072
COMMAND ===>                                            SCROLL ===> HALF
****** ************************ TOP OF DATA ************V10L30********
000100       SUBROUTINE MAKEM(A,N)
000200       REAL*8 A(N,N)
000300       DO 10 I=1,N
000400       DO 20 J=1,N
000500       A(I,J)=1.0D0/DBLE(I+J-1)  ! HILBERT MATRIX
000600 20    CONTINUE
000700 10    CONTINUE
000800       RETURN
000900       END
****** *********************** BOTTOM OF DATA ***********************
```

```
EDIT --- A79999A.TEST.FUNC.FORT(EX2) ------------------- COLUMNS 001 072
COMMAND ===>                                             SCROLL ===> HALF
****** ************************ TOP OF DATA ************V10L30********
000100       SUBROUTINE CSUM(Y,S,N)
000200       REAL*8 Y(N),S
000300       S=0.0D0
000400       DO 30 I=1,N
000500 30    S=S+Y(I)
000600       RETURN
000700       END
****** *********************** BOTTOM OF DATA ************************
```

サブルーチンライブラリ TEST.FUNC.FORT のメンバ EX1，EX2 を Fortran77 EX で翻訳して私用ライブラリ TEST.LOAD に登録する。サブルーチン名として書かれた MAKEM, CSUM が TEST.LOAD のメンバ名として登録される。翻訳、リンクのコマンドは、繁雑に打ち込む場合が多いので、一回一回 READY モードで打ち込むのは面倒である。そのため、コマンドを順番に実行するコマンドリストの列を作成しておくと、少しの修正で済む。

例では TEST.CLIST にコマンド列を作成し、 EXEC コマンドでライブラリの登録を行なう。

```
EDIT --- A79999A.TEST.CLIST ---------------------------- COLUMNS 009 080
COMMAND ===> EX ↵                                        SCROLL ===> HALF
****** ************************ TOP OF DATA ************V10L30********
000100   FORT TEST.FUNC.FORT(EX1) OBJ(TEST.OBJ(EX1)) NAME
000200   LINK TEST.OBJ(EX1) LO(TEST.LOAD) NCAL
000300   FORT TEST.FUNC.FORT(EX2) OBJ(TEST.OBJ(EX2)) NAME
000400   LINK TEST.OBJ(EX2) LO(TEST.LOAD) NCAL
000500   DEL TEST.OBJ
****** *********************** BOTTOM OF DATA ************************
```

```
Fortran77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
** MEMBER NAME **  MAKEM    NOW REPLACED IN LIBRARY.
Fortran77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
** MEMBER NAME **  CSUM     NOW REPLACED IN LIBRARY.
KQC0550I ENTRY (A) A79999A.TEST.OBJ DELETED
***
```

Fortran 77 EX/VP で翻訳する場合は、 FORT コマンドに VP オプションを追加する。ただし、 TSS からの実行は出来ない (0C6 アベンドする) ので、最終的なデバッグが完了してから VP での実行用のサブルーチンとして改めて登録を行なうこと。

最後に、作成した私用ライブラリを用いてメインプログラムを実行し、動作を確認する。RUN コマンドに LIB オペランドで私用ライブラリ名を指定して実行する。もちろん、 PFD/EDIT 内から RUN サブコマンドとしても実行可能である。

```
 READY
R MAIN.FORT LIB(TEST)  ↵
 Fortran77 EX COMPILER ENTERED
 END OF COMPILATION,HIGHEST SEVERITY CODE=00
 SOLUTION =  4930.93465695
 END OF GO,HIGHEST SEVERITY CODE=00
 READY
```

6.3 Step 3 : バッチジョブの依頼

TSS でデバッグが完了すると、ジョブ制御文を記述し、バッチジョブを VP に依頼する。VP 上での実行では、必ずユーザ自身で STEP パラメータと最適化レベルを指定すること。

- Step 1 で作成したプログラム TEST.FORT をバッチで実行する。ジョブクラスは CLASS=A, STEP は STEP=CLG、最適化レベルは OPT=E とする。ジョブ制御文は TEST.CNTL に作成する。ジョブ制御文を書くデータセットには必ず‘.CNTL’をつけること。 PFD/EDIT 内から SUBMIT サブコマンドでバッチジョブを依頼する。

```
EDIT --- A79999A.TEST.CNTL ---------------------- COLUMNS 001 072
COMMAND ===> SUB  ↵                                SCROLL ===> HALF
****** ********************** TOP OF DATA **********V10L30********
000100 //A79999AA JOB CLASS=A
000200 //    EXEC FORT,VP=YES,STEP=CLG,OPT=E
000300 //FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
000400 //
****** ******************** BOTTOM OF DATA ***********************
```

```
KEQ56208I ***           A79999AA        : (RECEIVED) ***
      *** A79999AA (J0998) A79999A : (JOB ACCEPTED) *** FIB CN(01)
KEQ56250I JOB A79999AA(JOB00998) SUBMITTED
***
```

ジョブの依頼が受け付けられると、上記のメッセージが表示される。

- Step 2 で作成したプログラム MAIN.FORT をバッチで実行する。ジョブクラスは CLASS=B, STEP は STEP=CLG、最適化レベルは OPT=B とする。私用ライブラリ TEST.LOAD を指定して実行する。 TEST.CNTL を次のように修正する。

```
EDIT --- A79999A.TEST.CNTL ---------------------- COLUMNS 001 072
COMMAND ===>                                      SCROLL ===> HALF
****** ********************** TOP OF DATA **********V10L30********
000100 //A79999AB JOB CLASS=B
000200 //    EXEC FORT,VP=YES,STEP=CLG,OPT=B,
000300 //     PRVLIB='A79999A.TEST.LOAD'
000400 //FORT.SYSIN DD DSN=A79999A.MAIN.FORT,DISP=SHR
000500 //
****** ******************** BOTTOM OF DATA ***********************
```

EXEC 文の継続はカンマ (,) で区切って改行した後に行なう。

6.4 ひとやすみ

バッチジョブの依頼が終了して、無事に実行中であることが確認できれば、TSS で他の作業が可能である。初歩的なミス (ジョブ制御文の書き間違い) の場合は、実行の前段階でシステムがジョブをつき返すので、あまりに処理が早い場合はエラーだと思って間違いない。

翻訳、実行に無事に移行した場合は、別のジョブを投入したり、他のアプリケーションを起動させたり出来る。もちろん、そのまま LOGOFF し、TSS セッションを終了して家に帰ってご飯を食べたり、デートに行ったりも自由である。結果は翌日セッションを開いて検索すればよい。

続け様にバッチジョブを連続投入すれば、システムを占有して利用できると思われるが、MSP には、ユーザの ID を認識して、出来るだけ公平にジョブを振り分ける機能があるので、そうはいかない。

確かに他人の課題を借りて、違う ID から連続投入すれば自分だけの計算が行なえるが、それはマナーの問題である。

6.5 Step 4 : バッチジョブの検索

依頼したバッチジョブの検索は `STATUS` コマンド、`SORP` コマンド、`PFD 3.8` コマンド、`OUTPUT` コマンドなどで行なう。例では、`MSO` コマンドを用いた出力検索を示す。MSO コマンドは PFD 画面から入力可能である。

```
EDIT --- A79999A.TEST.CNTL ---------------------- COLUMNS 001 072
COMMAND ===> MSO ↵                                SCROLL ===> HALF
****** ********************** TOP OF DATA **********V10L30********
000100 //A79999AB JOB CLASS=B
000200 //    EXEC FORT,VP=YES,STEP=CLG,OPT=B,
000300 //     PRVLIB='A79999A.TEST.LOAD'
000400 //FORT.SYSIN DD DSN=A79999A.MAIN.FORT,DISP=SHR
000500 //
****** ******************** BOTTOM OF DATA ***********************
```

```
---------------------------<  M S O  V02/L02  >--------------- BY KYUSHU UNIV.
                                                                      93.05.27
                    YOUR JOBS ARE AS FOLLOWS.
  (NO.) (JOBNAME)       (STATUS) (RANK) (NO.) (JOBNAME)       (STATUS) (RANK)
      1  A79999AB (J1005) OUT
   V  2  A79999AA (J1023) EXEC




       SEQ-NO.   ==>  1                         SORP-DSPRINT(PF5) DATASET
       OPTION    ==>        < B,NB,E,NE,C >          ==>
       NEW CLASS ==>        < H,O,K,S,E,U OPR-NO >
       TSS       ==>

-------------------------------------------------------------------------------
               PF1 ==> HELP  PF2 ==> RE-FRESH  PF3 ==> END
```

（NO.）項で V とあるのは、 VP 上で実行中の意味である。ジョブ番号 A79999AB のジョブは既に処理を終えて、出力待ちである。ジョブ番号 A79999AA のジョブは現在実行中である。PF2 キー（パソコンの場合は PF2 に割り当てられたキー）を押すと、画面がリフレッシュされ、最新のジョブ処理状態が表示される。

```
---------------------------<  M S O  V02/L02  >--------------- BY KYUSHU UNIV.
                                                                      93.05.27
                    YOUR JOBS ARE AS FOLLOWS.
  (NO.) (JOBNAME)       (STATUS) (RANK) (NO.) (JOBNAME)       (STATUS) (RANK)
      1  A79999AA (J1023) OUT
      2  A79999AB (J1005) OUT




       SEQ-NO.   ==>  1                         SORP-DSPRINT(PF5) DATASET
       OPTION    ==>  B     < B,NB,E,NE,C >          ==>
       NEW CLASS ==>        < H,O,K,S,E,U OPR-NO >
       TSS       ==>

-------------------------------------------------------------------------------
               PF1 ==> HELP  PF2 ==> RE-FRESH  PF3 ==> END
```

ジョブ番号 A79999A のジョブを検索する。 MSO 画面のオプションにカーソルを移動させ

B を入力する。

```
 BROWSE - A79999A.SSPEX.@001.D93117.T103419 ------------ LINE 00000 COLS 001
 COMMAND ===>                                                  SCROLL ===> CUR
********************************** TOP OF DATA *********************-CAPS ON-*
                      J E S   J O B  L O G  --  S Y S T E M  S P E X  --  N O
10.22.23 JOB 1023  KDS40613I USER(A79999A) LAST ACCESS DATE(1993.05.27),TIM
10.22.23 JOB 1023          *** A79999AA (J1023) A79999A : START    TIME=10.
10.22.32 JOB 1023  CD=0000 *** A79999AA (J1023) A79999A : END      TIME=10.

                           <<<  JCL STATEMENTS LIST  >>>    DATE  04/27/93

   1     //A79999AA JOB CLASS=A
   2     //    EXEC FORT,VP=YES,STEP=CLG,OPT=E
   8     //FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR

                            <<<  SYSTEM MESSAGES LIST  >>>

KDS40613I USER(A79999A) LAST ACCESS DATE(1993.05.27),TIME(10:08:50)
JDJ142I A79999AA FORT - STEP WAS EXECUTED - COND CODE 0000
JDJ373I STEP/FORT    / START  93117.1022
JDJ374I STEP/FORT    / STOP   93117.1022 CPU      0MIN 00.26SEC  SRB      0
ACTRMSG STEP/FORT    /                   VPU      0MIN 00.00SEC
JDJ142I A79999AA LKED - STEP WAS EXECUTED - COND CODE 0000
JDJ373I STEP/LKED    / START  93117.1022
                              :
                              :
```

検索は PFD/EDIT の要領で行なう。終了は PF3 キー。検索を終えて、プリンター等に出力する必要のない場合は、 OPTION 覧で C を指定することでキャンセルされ、ジョブはシステムから消去される。

6.6 Step 5 : バッチジョブの出力

実行が終了したバッチジョブは、センターのプリンターに出力可能である。また、データセットへ保存することも出来る。紙の節約の観点から、 MSO 等でジョブを検索した後に、必要なジョブのみを出力することをお勧めする。

6.6.1 プリンターへの出力

九州大学大型計算機センターのプリンターはオープン機器室に設置された日本語ラインプリンター (NLP) とカット紙ラインプリンター (CLP)、及びクローズド機器室に設置された B4 版, A4 版の NLP とが利用可能である。

ジョブの出力をプリンターに行なうには、出力要求用のコンソールから出力を依頼する。先ず、ジョブが終了していることを STATUS コマンドで確認する。

```
 READY
ST ⏎
 KEQ56211I JOB A79999AW(JOB01084) IS EXECUTING
 KEQ56211I JOB A79999A#(TSU05318) IS EXECUTING ON THIS TERMINAL
 KEQ56211I JOB A79999A#(TSU05319) IS EXECUTING
 KEQ56192I JOB A79999AA(JOB01099) IS WAITING FOR OUTPUT
 KEQ56192I JOB A79999AB(JOB01100) IS WAITING FOR OUTPUT
 READY
```

A79999AA, A79999AB は実行を終了し、出力待ちである。 A79999AW は現在実行中である。

計算結果は一旦システム内に保存されている。出力要求用のコンソールから登録番号とパスワードを入力すると、出力可能なジョブの一覧が表示される。ジョブ名の先頭に O を指定することによりプリンターへの出力が行なわれる。クローズドラインプリンター(センター2階ロータリーテーブル横)の場合は、ジョブ名の先頭に H を指定することにより、 A4 版での高速出力が出来る。プリンターに出力されると、出力結果はシステムから消去される。プリントアウトした後で MSO コマンド等による検索は出来ないので注意。

なお、バッチジョブの出力結果は 2 週間を過ぎるとシステムから消去されるのでその期間までに検索・出力を行なうこと。

6.6.2 データセットへの保存

バッチジョブの出力結果は、データセットに保存することも可能である。データセットへの書き出しは OUTPUT コマンドで行なう。ジョブ A79999AA をデータセット TEST.OUTLIST に出力する例を挙げる。

```
 READY
OUT A79999AA PR(TEST) ⏎
 READY
```

データセット名として PR(TEST) と指定すると、勝手に .OUTLIST が最後に付加される。これを抑止したい場合は完全データセット名を指定する。

```
 READY
OUT A79999AB PR('A79999A.TEST.DATA') ⏎
 READY
```

データセットに出力されると、ジョブはシステムから消去されるので注意。また、出力結果が巨大な場合、データセットへの出力中、省略値で定められた出力データセットの量を越えてしまい、結果が途中までしか出力できない場合がある。その場合は予め ALLOCATE コマンドまたは PFD のオプションで大きめの容量を確保したデータセットを作成した上で、そのデータセットに対してジョブの出力を行なう。

第 II 部
応用編

7　システム記憶 (SSU)

SSU は System Storage Unit の略称であり、記憶装置の一種である。 VP の演算において使用される記憶装置としては、主記憶装置（いわゆるメモリー）と磁気ディスク装置がある。主記憶装置は、処理が高速に行なえるが高価であり、磁気ディスク装置は大容量のデータが保存できるが処理に時間がかかる。 SSU は拡張記憶装置として、これらの中間的な位置を占める半導体記憶装置である。

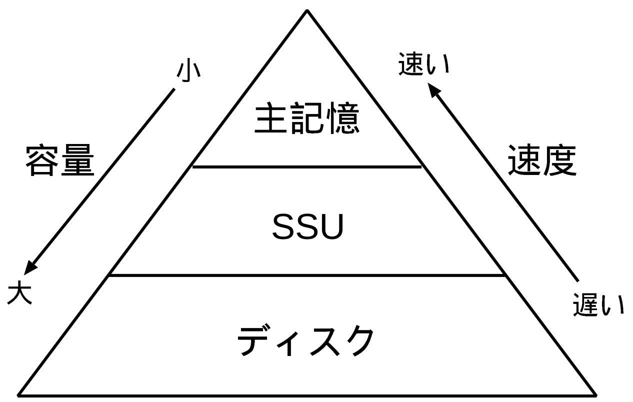


図 3: 記憶装置の位置づけ

7.1　SSU の特徴

SSU の特徴として、以下の点が挙げられる。

- VP 上で最大 500MB の SSU が確保できる。 500MB は Fortran プログラムの入出力用の作業ファイルとして割り当てることが可能である。また、 Fortran プログラムの仮想的な主記憶として、配列の確保も可能である。このため、主記憶の最大領域 500MB と合わせて、合計 800MB の領域が使用できる。

- ただし、 SSU を配列として用いた場合（SSU 配列）、 SSU 配列のデータは、一度主記憶に転送されて演算を行ない、再び主記憶を介して SSU にデータが格納される機能になっている。そのため、主記憶と SSU とのデータ転送を上手に行なわないと、演算時間が大幅に増加する。データ転送を効率良く行なうためのチューニングは、ユーザの Fortran プログラミング技術に大きく依存する。

- SSU はバッチ処理のみ使用できる。 TSS では使用できない。

- VP で SSU の使用できるジョブクラスは V である。ジョブクラス A, B では SSU の使用が出来ない。

- SSU の使用できる領域は、 Fortran の入出力ファイルと、 SSU 配列とに同時の割り当てが可能である。ただし、使用領域の合計が 500MB を越えた場合はジョブがキャンセルされる。

- 500MB を越えて SSU を使用したい場合は、要審査ジョブ (D ジョブ) としてセンターに申し込むこと。

7.2 SSU の作業用ファイルとしての利用

Fortran プログラムでは、作業用ファイルとして一時的に入出力を大量に行なう場合がある。作業用ファイルの割り当てには、ディスクに割り当てる方法と、主記憶に割り当てる方法がある。（主記憶または SSU に割り当てられた作業用ファイルを VIO/F ファイルと呼ぶ）。ディスクに割り当てた場合は、大容量の作業領域が確保されるが、演算時間が長時間にわたる。主記憶に割り当てた場合 (VIO/F ファイル) は、実行時間が短縮される一方で領域の確保が配列と競合していた。 SSU を VIO/F ファイルとして使用することで、主記憶は最大 300MB 確保したままで、更にディスクに比べてはるかに高速に処理が行なえる作業用ファイルが最大 400MB 使用できる。

7.2.1 VIO/F ファイル装置の指定方法

VIO/F ファイルとして SSU を使用するには、ジョブ制御文の中で VIO/F ファイルの DD 文で `UNIT=SSU` を指定し、 `SPACE` パラメータで SSU の容量を指定する。

【使用例】

- 装置機番 1 番 `READ(1,*), WRITE(1,*)` を作業用ファイルとして使用する。作業用ファイルを `SSU` に `30MB` 割り付ける。

```
//GO.FT01F001 DD SUBSYS=(VPCS,'SPACE=30M'),UNIT=SSU,
//            DISP=(NEW,DELETE)
```

- 装置機番 `18` 番 `READ(18,*), WRITE(18,*)` に `100MB`、装置機番 `99` 番 `READ(99,*), WRITE(99,*)` に `75MB`、各々作業用ファイルを `SSU` に割り付ける。

```
//GO.FT18F001 DD SUBSYS=(VPCS,'SPACE=100M'),UNIT=SSU,
//            DISP=(NEW,DELETE)
//GO.FT99F001 DD SUBSYS=(VPCS,'SPACE=75M'),UNIT=SSU,
//            DISP=(NEW,DELETE)
```

7.2.2 プログラム例

入出力ファイルを使用したプログラム例を挙げる。

```
EDIT --- A79999A.SSU.FORT(EX1) ------------------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** ***************************** TOP OF DATA ****************V10L30******
000100       PARAMETER(N=1000)
000200       REAL*8 A(N),X(N)
000300
000400       DO 10 I=1,N
000500 10     A(I)=DBLE(I)
000600
000700 C--------------- DISK UNIT ------------------------------
000800       WRITE(10) A
000900       REWIND(10)
001000       READ(10) X
001100
001200 C--------------- SYSTEM STORAGE UNIT ---------------------
001300       WRITE(11) A
001400       REWIND(11)
001500       READ(11) X
001600
001700 C--------------- MAIN STORAGE UNIT -----------------------
001800       WRITE(12) A
001900       REWIND(12)
002000       READ(12) X
002100       END
****** **************************** BOTTOM OF DATA **************************
```

SSU.FORT(EX1) は、論理機番 10, 11, 12 に、全く同じデータの読み書きを行なうだけのプログラムである。次に、ジョブ制御文で機番 10, 11, 12 にディスク、 SSU、主記憶を各々割り当てる。

```
EDIT --- A79999A.SSU.CNTL(EX1) ------------------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** ***************************** TOP OF DATA ****************V10L30******
000001 //A79999A1 JOB CLASS=V
000002 //    EXEC FORT,VP=YES,STEP=CLG,OPT=B
000003 //FORT.SYSIN  DD DSN=A79999A.SSU.FORT(EX1),DISP=SHR
000004 //GO.FT10F001 DD DSN=&TEMP1,UNIT=WORK,DISP=(,DELETE),
000005 //      SPACE=(23476,(2500,100)),
000006 //      DCB=(RECFM=VBS,BLKSIZE=23476,BUFNO=1)
000007 //GO.FT11F001 DD SUBSYS=(VPCS,'SPACE=50M'),UNIT=SSU,DISP=(,DELETE)
000008 //GO.FT12F001 DD SUBSYS=(VPCS,'SPACE=50M')
000009 //
****** **************************** BOTTOM OF DATA **************************
```

一時データセットに対する入出力を高速化するために SSU を使って入出力処理を行う方法では、対象となる入出力文は、書式なし順次 / 直接入出力文、ファイル位置づけ入出力文、OPEN 文、 CLOSE 文で、主記憶ファイルを DD 名で指定する。

7.3 SSU の Fortran 配列としての利用

Fortran の名前付き共通ブロックに属する COMMON 配列を SSU に割り当てることで、SSU を仮想的な主記憶として用いることが出来る。これにより主記憶 300MB、 SSU 500MB、合計 800MB の大容量記憶空間が使用できる。但し、 SSU 配列のデータは一度主記憶装置 (MSU) に転送した後に演算を行ない、演算結果が MSU を経由して SSU に格納される仕組みになっている。 Fortran システムは、自動的に MSU と SSU とのデータ転送を行なうが、 Fortran プログラムにおいて、データ転送を意識したチューニングを行なうことで、大幅な実行時間の短縮が可能である。プログラム作成のコツは [3] を参照。

7.3.1 SSU 配列の指定方法

SSU 配列として SSU を使用するには、ジョブ制御文の中で実行ステップの SSARRAY の DD 文で `UNIT=SSU` を指定し、 `SPACE` パラメータで SSU の容量を指定する。

【使用例】

- `SSU` 配列を `200MB` 使用する。

```
//GO.SSARRAY DD SUBSYS=(VPCS,'SPACE=200M'),UNIT=SSU,
//         DISP=(NEW,DELETE)
```

- `SSU` の使用容量の指定が `400MB` 以内ならば、 `VIO/F` ファイルと併用できる。

```
//GO.FT09F001 DD SUBSYS=(VPCS,'SPACE=75M'),UNIT=SSU,
//          DISP=(NEW,DELETE)
//GO.SSARRAY DD SUBSYS=(VPCS,'SPACE=300M'),UNIT=SSU,
//          DISP=(NEW,DELETE)
```

Fortran の名前付き共通ブロックに属する COMMON 配列は、翻訳オプションで `SSU(共通ブロック名)` と指定する。

```
//A79999A7 JOB CLASS=V
//  EXEC FORT,VP=YES,STEP=CLG,OPT=E,
//      OPTION='SSU(共通ブロック名)'
//FORT.SYSIN  DD DSN=A79999A.VP.FORT,DISP=SHR
//GO.SSUARRAY DD SUBSYS=(VPCS,'SPACE=200M'),
//      UNIT=SSU,DISP=(NEW,DELETE)
//
```

7.3.2 プログラム例

```
EDIT --- A79999A.SSU.FORT(EX2) ------------------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** ***************************** TOP OF DATA **************V10L30********
000100       PROGRAM SSU
000200       PARAMETER(N=1000)
000300       REAL*8 A(N,N),B(N)
000400       COMMON/SSUA/A         ! SSU ARRAY
000500
000600       DO 10 I=1,N
000700 10    B(I)=DBLE(I)
000800
000900       DO 20 I=1,N
001000       DO 30 J=1,N
001100       A(I,J)=B(I)*B(J)
001200 30    CONTINUE
001300 20    CONTINUE
001400       END
****** **************************** BOTTOM OF DATA **************************
```

SSU 配列として 2 次元配列 A を COMMON 文で宣言する。プログラミングに関する注意は [3] を参照。

ジョブ制御文では翻訳オプションとして SSU(共通ブロック名) を指定する。また、 SSARRAY の DD 文で SSU の使用容量を MB 単位で指定する。例題では共通ブロック名として SSUA としたが、 8 文字以内で適当に定める。

```
EDIT --- A79999A.SSU.CNTL(EX2) ------------------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** ***************************** TOP OF DATA **************V10L30********
000001 //A79999AZ JOB CLASS=V
000002 //    EXEC FORT,VP=YES,STEP=CLG,OPTION='SSU(SSUA)',OPT=E
000003 //FORT.SYSIN DD DSN=A79999A.SSU.FORT(EX2),DISP=SHR
000004 //GO.SSARRAY DD SUBSYS=(VPCS,'SPACE=50M'),UNIT=SSU,DISP=(,DELETE)
000005 //
****** **************************** BOTTOM OF DATA **************************
```

8 高速入出力機能

主記憶領域に収まらない巨大データを扱いたい場合、また、演算の中間結果が巨大化し、主記憶領域に常駐出来ない場合、高速入出力機能を使用する。

高速入出力機能は、データの入出力を高速化するために、入出力バッファ (一時的に使用される記憶領域) を使わず、複数台の DASD(直接アクセス記憶装置) に並列に入出力処理を行う方法である。対象となる入出力文は、書式なし順次入出力文、 REWIND 文、 OPEN 文、 CLOSE 文で、入出力装置は DD 文で指定する。スペースの確保はシリンダ単位で行なう。詳細は [4] を参照。

ファイル容量が少ない場合は、主記憶ファイル入出力や SSU の VIO/F 機能を用いた方が処理が高速に実行される。

【指定できる機番】

FTnnS001 から FTnnSmmm まで

nn - 装置参照番号

mmm - 並列処理数

'S' は高速入出力機能を使用することを示す。FTnnSmmm の指定がない場合や、FTnnF001 と FTnnSmmm の両方を指定した場合は、高速入出力機能は使用できない。

8.1 プログラム例

高速入出力機能で出力したデータは、高速入出力機能で入力する。複数台の入出力装置に並列処理する場合、入力と出力は同じ大きさで行なう。

```
EDIT --- A79999A.SSU.FORT(EX3) ------------------------------ COLUMNS 001 072
COMMAND ===>                                                  SCROLL ===> HALF
****** ***************************** TOP OF DATA **************V10L30********
000100       REAL*8 A(10000),B(4000),C(1000),D(5000),E(5000),F(5000)
000200
000300       DO 10 I=1,10000
000400 10    A(I)=DBLE(I)
000500
000600       DO 20 I=1,4000
000700 20    B(I)=1.0D0/DBLE(I)
000800
000900       DO 30 I=1,1000
001000 30    C(I)=2.0D0*DBLE(I)
001100
001200       WRITE(10) A
001300       WRITE(10) B,C
001400       REWIND(10)
001500       READ(10) D,E
001600       READ(10) F
001700       END
****** **************************** BOTTOM OF DATA **************************
```

一時データセットの入出力を 3 台の WORK ボリューム（WORK1 ~ WORK6 まで指定可能）を使って並列処理を行なう。

```
EDIT --- A79999A.SSU.CNTL(EX3) ----------------------------- COLUMNS 001 072
COMMAND ===>                                                 SCROLL ===> HALF
****** ***************************** TOP OF DATA **************V10L30********
000001 //A79999AA JOB CLASS=B
000002 //    EXEC FORT,VP=YES,STEP=CLG,OPT=B
000003 //FORT.SYSIN  DD DSN=A79999A.SSU.FORT(EX3),DISP=SHR
000004 //GO.FT10S001 DD DSN=&TEMP1,UNIT=WORK1,DISP=(,DELETE),
000005 //      SPACE=(CYL,(5),,CONTIG)
000006 //GO.FT10S002 DD DSN=&TEMP2,UNIT=WORK2,DISP=(,DELETE),
000007 //      SPACE=(CYL,(5),,CONTIG)
000008 //GO.FT10S003 DD DSN=&TEMP3,UNIT=WORK3,DISP=(,DELETE),
000009 //      SPACE=(CYL,(5),,CONTIG)
000010 //
****** **************************** BOTTOM OF DATA **************************
```

9 主記憶ファイル入出力機能

一時データセットに対する入出力を高速化するために、DASD の代わりに主記憶装置を使って入出力処理を行う方法がある。対象となる入出力文は、書式なし順次 / 直接入出力文、ファイル位置づけ入出力文、OPEN 文、CLOSE 文で、主記憶ファイルを DD 名で指定する。この領域は拡張リージョンに確保され、複数指定も可能だが、領域の合計がリージョンサイズの制限値以内でなければならない。スペースの単位はメガバイト（MB）で、最大 300MB まで指定できる。

```
//A79999A8 JOB CLASS=V
//  EXEC FORT,VP=YES,STEP=CLG,OPT=B
//FORT.SYSIN  DD DSN=A79999A.VP.FORT,DISP=SHR
//GO.SYSIN    DD DSN=A79999A.VP.DATA,DISP=SHR
//GO.FT10F001 DD SUBSYS=(VPCS,'SPACE=100')
//
```

使用例は SSU の VIO/F 機能と同様なので省略。

10 マスストレージディスク

センターでは、100MB 単位でマスストレージディスクの貸し出しを行っている。保存媒体は MTL (磁気テープライブラリ装置)。一登録番号につき、100MB を 1 本として 5 本まで貸し出している。料金は 1 本につき 1 カ月 1000 円である。

貸し出しは、支払責任者が 1 名いればグループでもよい。この場合、マスストレージ内のファイルは共用になる。なお、登録後は取消申請届か登録番号が無効になるまで使用できる。登録手続きや使用法は九州大学大型計算機センター広報の Vol.24, No.3, pp.284–286, 1991. を参照。

11　ベクトルユニット使用時間の測定

VP2600 での Fortran コンパイラには、ベクトルユニット (vector unit : VU) の使用時間を計測することで、ベクトル化率を計測する CLOCKV サブルーチンがサポートされている。 CLOCKV サブルーチンを用いて DO ループ単位、サブルーチン単位の CPU 時間および VU 時間 (ベクトルユニットの使用時間) が計測できる。これにより、ベクトル化による性能向上の程度が判断できる。なお、 CPU 時間のみの測定には CLOCK, CLOCKM サブルーチンがある (cf.[4])。

【引用形式】

```
CALL CLOCKV(G1,G2,I1,I2)
```

G1 : プログラム実行開始からの VU 時間
G2 : プログラム実行開始からの CPU 時間
I1 : 計測時間の単位を指定する。省略値は 0
- 0 – 秒単位
- 1 – ミリ秒単位
- 2 – マイクロ秒単位

I2 : 計測時間を返却する値の型を指定。省略値は 0
- 0 – 整数
- 1 – 実数
- 2 – 倍精度実数

【使用例】

- CLOCKV サブルーチンを用いて、ある単位 (DO ループ) のベクトルユニット時間、および CPU 時間を測定する。

```
      SUBROUTINE TESTSUB(A,B,N)
       REAL*8 A(1001,1001),B(1001,1001)
        CALL CLOCKV(VU1,CPU1,0,1)  !   START
          DO 10 I=1,N
          DO 10 J=1,N
           A(J,I)=A(J,I)+B(J,I)
   10     CONTINUE
        CALL CLOCKV(VU2,CPU2,0,1)  !   END

        VUTIME  =  VU2 - VU1
        CPUTIME = CPU2 - CPU1
        WRITE(6,*) 'VU  TIME = ',VUTIME,'SECONDS '
        WRITE(6,*) 'CPU TIME = ',CPUTIME,'SECONDS'
      END
      RETURN
```

12 Fortran77 EX/VP の翻訳オプション

Fortran プログラムの VP 上での実行でよく用いられる翻訳オプションについて簡単に説明する。詳細は [4], [5] を参照。

12.1 DEBUG オプション

デバッグのための検査を行なう。 DEBUG と指定し、括弧（ ）内に検査を行ないたい機能を指定する。なお、本オプションを指定すると、実行時間が大幅にかかる場合があるので、注意すること。

- 引数の妥当性の検査（ARGCHK）
- 添字式及び部分列式の値の検査（SUBCHK）
- 未定義データの引用の検査（UNDEF）
- 整数型オーバーフローの検出（IOVERFL）
- 重複した実引数の副プログラム内での定義検査（OVERLAP）

【使用例】

- 引数の検査を行なう。

```
//  EXEC FORT,STEP=CLG,VP=YES,OPT=E,OPTION='NOADV,DEBUG(ARGCHK)'
```

- 添字式の確認、整数型オーバーフローの検査を行なう

```
//  EXEC FORT,STEP=CLG,VP=YES,OPT=E,
//        OPTION='NOADV,DEBUG(SUBCHK,IOVERFL)'
```

12.2 FLAG オプション

翻訳メッセージの出力レベルを指定する。省略値では、全ての翻訳診断メッセージが出力される。 FLAG(W) で I レベルのメッセージが抑止される。また、 FLAG(S) で I, W レベルのメッセージが抑止される。 I はエラーではないが、注意を促すメッセージ、 W は処理が続行可能な軽度のエラーを表す。

【使用例】

- I, W レベルの翻訳診断メッセージの出力を抑止する。

```
//  EXEC FORT,VP=YES,OPT=E,OPTION='FLAG(S)'
```

12.3 最適化に関するオプション

基本のオプションは OPT である。 INLINE, XOPT によってレベルの調節が可能。また、OPTMSG によって最適化のメッセージが出力される。

- 基本的な最適化 (OPT=B)
- 拡張最適化 (OPT=E)
- 最大限の最適化 (OPT=F)

【使用例】

- 最大限の最適化を行ない、高速な実行を計る。最適化のメッセージを出力させ、最適化に伴う副作用の有無を確認する。

```
//  EXEC FORT,VP=YES,OPT=F,OPTION='OPTMSG'
```

カタログドプロシジャ FORT では特に最適化オプションを重要として、 OPTION= の指定をせず、 OPT= の指定で最適化レベルが設定可能になっている。
上記例題と OPTION='OPT(F),OPTMSG' とは同様である。

12.4 NOPRINT オプション

リスト情報を出力しないことを指定。デバッグが完了して、結果だけが知りたいのに、プログラムリストや、各サブルーチンの割り付けリストや、ベクトル化メッセージ等がたくさん出力され、イライラする時は、このオプションを指定する。システムのログに続いてすぐに実行結果が出力され、すっきりする。

【使用例】

- リスト情報を出力しないことを指定。

```
//  EXEC FORT,VP=YES,OPT=E,OPTION='NOPRINT'
```

翻訳時の印刷情報は、翻訳オプション FLAG, NOSOURCE, STATIS 等で細かく指定できる。詳しくは [4] を参照。

13 アナライザの利用

アナライザは、 Fortran プログラムの動作情報を収集および分析して、チューニング作業（ベクトル計算機上での実行性能を改善すること）を支援するツールである。 VP2600 上で Fortran プログラムは、 Fortran77 EX/VP コンパイラで翻訳されることによって、ベクトル化（ループ内の演算がベクトル命令に自動的に変換）され、実行が高速に行われる。この Fortran プログラムに、ベクトル化される部分が多いほど処理が高速化されることになる。プログラムのチューニング作業の目標として、ベクトル化される部分を出来るだけ多くすることが挙げられる。一般的に、プログラム中で実行時間が多くかかる部分は、ある特定の場所に偏る場合が多い。従って、その部分を集中的に改善（ベクトル化）することが高速な処理への最短距離である。アナライザはその情報として、実行頻度、実行回数、実行時間、コスト値[6]などの動作情報を提供するツールである。

13.1 アナライザの機能

アナライザには次の 2 つの機能がある。

サンプリング解析機能

プログラムを実際に実行して、プログラム全体の動作状況を大まかに解析する機能。具体的には、プログラムのルーチン単位ごとに、実行頻度（コスト値）の分布を計測する。これより、プログラムのどのルーチンが最も実行時間がかかるかを把握することが出来る。

詳細解析機能

実行文の実行回数やコストなどにより、プログラムの動作を詳細に解析する機能。詳細解析機能は、更に 2 つの機能を持つ。

1. 見積り詳細解析機能
   プログラムを模擬的に実行し、その動作状況を詳細に解析する機能。この機能は、実行時に決定されるプログラムの実行条件（ループの繰返し回数など）を仮定してプログラムを模擬的に実行する。結果はルーチン単位や文単位ごとに実行回数やコスト値などを計測して出力する。この機能により、各ルーチンの中でどの部分が最も実行時間を費やすかを予測することが出来る。見積り詳細解析は、翻訳情報だけを使用してプログラムのコスト分布を解析する。実行条件として以下を仮定している。

   - ループの繰返し回数は、繰返し回数が定数であったり、最適化制御行での指定があるなど、明らかに回数が分かる場合は、その値を仮定する。明らかでないときには、ループ内で使用されている配列のサイズ、または 100 を仮定する。
   - IF 文の真率は、最適化制御行での指定がある場合には、その値であると仮定する。そうでない場合は、 50% を仮定する。

[6] コスト値とは、プログラムの実行のためのおよその費用（実行時間を予測するための相対的な値）のこと。例えば、演算数が少なかったり、演算が単純であればコスト値は低く、関数の評価などを含む複雑な文はコスト値が高くなる。

2. 実行詳細解析機能
プログラムを実際に実行して、その動作状況を詳細に解析する機能。出力はルーチン単位ごとや文単位ごとに、実行回数、コスト値や CPU 時間などを計測して出力する。この情報から、プログラム全体のコスト分布を予想したり、各ルーチン中のどの部分で最も実行時間を費やしているかを把握することが出来る。実行詳細解析機能は、翻訳情報と実行情報の両方を使用してプログラムのコスト分布を解析して出力する。

13.2 アナライザ機能の使い分け

1. アナライザは、バッチジョブのみの動作であり、 TSS ジョブでは動作しない。また、アナライザは汎用計算機（M1800）およびベクトル計算機（VP2600）で実行が可能である。プログラムの解析目的は、最終的にはベクトル計算機上での高速な実行であるが、そのための情報は汎用機、ベクトル計算機どちらでも収集ができる。

2. サンプリング解析機能は、主にプログラム全体の動作状況を把握して、実行コストのかかる部分を検出するための機能である。詳細解析機能は、コストの高い部分をより詳細に分析・評価するための機能である。

13.3 アナライザの起動方法

アナライザはバッチ処理のみの起動である。また、汎用機 (M1800)、ベクトル計算機 (VP2600) の双方で実行可能。カタログドプロシジャ ANALYZE は以下の通り。

	主なパラメータ
ANALYZE	,DSN=' ソースデータセット名' [, MODE=SAMP] [, ANAOPT=' アナライザオプション'] [, OPTION=' コンパイラオプション'] [, PRVLIB=' データセット名'] [, VP=YES] [, VREGION= リージョンサイズ]

【記号パラメータの説明】

DSN= : 解析を行うソースプログラムのデータセット名を指定。
MODE=SAMP : サンプリング解析機能を行うことを指定。省略時は詳細解析機能。
ANAOPT : アナライザオプションを指定。
OPTION : Fortran77 EX/VP コンパイラオプションを指定。
PRVLIB : 私用ライブラリを指定。
VP=YES : VP2600 で実行することを指定、省略時は M1800 で実行される。
VREGION : VP=YES のとき VP のリージョンサイズを指定。
省略時は各ジョブクラスの制限値。

【使用例】

- ベクトル計算機 VP2600 上でサンプリング解析を行う。ソースプログラム TEST.FORT, 使用データセット TEST.DATA。

```
//A79999AA JOB CLASS=A
//  EXEC ANALYZE,DSN='A79999A.TEST.FORT',MODE=SAMP,VP=YES
//ANA.FT05F001 DD DSN=A79999A.TEST.DATA,DISP=SHR
//
```

- ベクトル計算機 VP2600 上でサンプリング解析を行う。ソースプログラム TEST.FORT、使用データセット TEST.DATA。サンプリング解析オプションとして実行頻度採取の時間間隔値を小さくして、計測精度を高める (cf.[20]) 。

```
//A79999AB JOB CLASS=B
//  EXEC ANALYZE,DSN='A79999A.TEST.FORT',MODE=SAMP,VP=YES,
//       ANAOPT='INTERVAL(1000)'
//ANA.FT05001 DD DSN=A79999A.TEST.DATA,DISP=SHR
//
```

  サンプリング解析機能は、実行時の振る舞いからコストを解析する。従ってプログラムの実行が実際に行われる。情報の見方は [19], [20] 参照。

- ベクトル計算機 VP2600 上で見積り詳細解析を行う。ソースプログラム TEST.FORT、私用ライブラリ TEST.LOAD。アナライザオプションとして STATIC を指定する。

```
//A79999AC JOB CLASS=V
//  EXEC ANALYZE,DSN='A79999A.TEST.FORT',VP=YES,
//       PRVLIB='A79999A.TEST.LOAD',ANAOPT='STATIC'
//
```

- ベクトル計算機 VP2600 上で実行詳細解析を行う。ソースプログラム TEST.FORT、アナライザオプションとして COUNT, VECTOR を指定する。

```
//A79999AD JOB CLASS=V
//  EXEC ANALYZE,DSN='A79999A.TEST.FORT',
//       ANAOPT='VECTOR,COUNT',VP=YES
//
```

- アナライザオプションに COUNT を指定し、アナライザ解析情報を収集する。解析情報の出力は、新規に作成するデータセット TEST.SYSPINF に行なう。

```
//A79999AE JOB CLASS=V
//  EXEC ANALYZE,DSN='A79999A.TEST.FORT',
//       ANAOPT='VECTOR,COUNT',VP=YES
//ANA.SYSPINF DD DSN=A79999A.TEST.SYSPINF,UNIT=PUB,
//    DCB=(RECFM=VB,LRECL=600,BLKSIZE=23440),
//    DISP=(NEW,CATLG),SPACE=(TRK,(10,10),RLSE)
//
```

  収集された解析情報は TSS でチューナを起動することで、呼び出される。チューナの起動方法及び実行は [18] を参照。

14 マニュアルのオンライン検索 (PLUM コマンド)

[4] は Fortran77 EX の使用手引書として、是非手元に欲しい一冊だが、かなり分厚い本であり、購入すると岩波の数学辞典よりも高い。また、計算機メーカーのマニュアルの常として非常に読みにくい。しかし、[4] はオンラインマニュアルとして端末から参照可能であり、キーワード検索を行なうことで、読みにくいマニュアルをめくる苦痛から解放される。

ただしパソコン端末の場合、エミュレーターによってオンライン検索が出来ない場合があるので、注意されたい。

オンラインマニュアルの起動は、TSS の `PLUM` コマンド で行なう。下図の様に、PFD の EDIT 機能で編集中の画面からも起動可能である。

```
EDIT --- A79999A.TEST.FORT(EX1) ----------------------------- COLUMNS 001 072
COMMAND ===> PLUM ↵                                              SCROLL ===> HALF
****** ***************************** TOP OF DATA **************V10L30********
000100       PROGRAM FUNC
000101       IMPLICIT REAL*8(P)
000110       PARAMETER(PI=3.14159265358979323846D0)
000200       REAL*8 Z/0.0D0/,T
000210 C     T=DSIN(PI*2.0D0)
000220       T=DSIN(PI)
000300       WRITE(6,*)'T= ',T
000700       DO 10 I=1,10000
000800        Z=Z+DSIN(PI*DBLE(I))
000900 10    CONTINUE
001000       WRITE(6,*)'Z= ',Z
001100       END
****** **************************** BOTTOM OF DATA **************************
```

PLUM コマンドを入力すると、フルスクリーン環境に切り替わり、メニューが表示される。

Fortran77 EX の利用方法を検索したい場合は、表示メニューで 5 番を指定し、ENTER キーを押す。Fortran77 EX のマニュアルが呼び出される。

```
                                                    XTERM6683
> █
    目的項目一覧                                        01/12
                                    COPYRIGHT FUJITSU LIMITED 1990
システムの概要

    1．ＦＯＲＴＲＡＮ７７　ＥＸシステムの概要

    2．翻訳から実行まで

ジョブ投入の手続き

    3．ジョブを起動する

    4．原始プログラムを翻訳する

    5．ロードモジュールを作成する

    6．ジョブ制御文を変更する

    7．翻訳／結合編集／実行する

                                                《次の画面に続く》
HELP=01 SPLIT=02 EXIT=03 PRIMARY=04 FORWARD=05 PRINT=06 BACKWARD=07 FORWARD=08
```

例として COS 関数の機能を検索してみる。コマンドプロンプト覧 (＞) で、'COS' と打ち込み、 ENTER キーを入力することでキーワード検索が行なわれる。画面下のファンクションキー (大体 PFD の割当と同じ) に従って、検索を進めると、最終的に次の様な COS 関数に関する情報にたどり着く。

```
                                                    XTERM6683
> █
    余弦関数                                            02/04
                                    COPYRIGHT FUJITSU LIMITED 1990
    実数上の余弦関数

関数値ｙの定義（続き）
               100
    ｜ａｒ｜≦２
    ｜ａｉ｜≦１７４．６７３

引数の型と関数の型

    総称名：ＣＯＳ
    ─────────────────────────────────────
      個別名    ｜引数の｜第１引数｜第２引数｜関数の型｜    関数値ｙの範囲
                ｜個　数｜の型    ｜の型    ｜        ｜
    ────────┼───┼────┼────┼────┼──────────
    ＣＯＳ      ｜  １  ｜Ｒ＊４  ｜   －   ｜Ｒ＊４  ｜   －１≦ｙ≦１
    ＤＣＯＳ    ｜      ｜Ｒ＊８  ｜   －   ｜Ｒ＊８  ｜        〃
    ＱＣＯＳ    ｜      ｜Ｒ＊１６｜   －   ｜Ｒ＊１６｜        〃
    ─────────────────────────────────────
    引数の型と関数の型の記法は，“組込み関数表の記法”に示す．
                                                《次の画面に続く》
HELP=01 SPLIT=02 EXIT=03 PRIMARY=04 FORWARD=05 PRINT=06 BACKWARD=07 FORWARD=08
```

終了は PF3 キー、または PF3 キーに割り当てられているキーで行なう。 PLUM が終了すると、もとの PFD/EDIT 画面に戻る。

このように、 MSP の TSS 機能を用いて Fortran プログラムを作成 / デバッグする時は、オンラインマニュアルで簡単に Fortran77 EX の機能が検索できる。操作方法はメニュー形式で行なう。とりあえず終了するには PF3 キーを連打すればよいことだけ覚えて、どんどん利用されたい。

15　メッセージのオンライン検索 (LM コマンド)

Fortran システムでは、翻訳・実行中にプログラムのエラーや実行時異常終了 (ABEND) 時にメッセージを出力する。 Fortran77 EX のメッセージおよび MSP システムが発行するシステムメッセージとシステムコードは TSS の LM コマンド で検索出来る。 LM コマンドの入力形式は次の通り。

LM メッセージ ID
LM メッセージ ID LINE　　（TTY 手順日本語端末の場合）

パソコン端末等から LM コマンドを入力する場合は、必ず LINE（ラインモードでの出力）を指定すること。 LINE の指定をしないと、メッセージが乱れて出力される。

【よく用いる LM コマンドの使用例】

前日 VP に長時間の Fortran ジョブを依頼していた。翌日実行が終了し、出力待ちになっていたので、実行結果を検索したところ、次のメッセージが冒頭に出力されていた。さて...?

```
IBROWSE - A79999A.SSPEX.@001.D93127.T111823 ------------ LINE 00000 COLS 001 08
 COMMAND ===>                                                  SCROLL ===> PAGE
********************************** TOP OF DATA ********************-CAPS ON-***
                     J E S   J O B  L O G -- S Y S T E M  S P E X -- N O D

10.01.45 JOB 1640  KDS40613I USER(A79999A) LAST ACCESS DATE(1993.04.28),TIME(09
10.01.45 JOB 1640          *** A79999AA (J1640) A79999A : START    TIME=10.01.4
20.19.29 JOB 1640  +JWE0019I-U THE PROGRAM WAS TERMINATED ABNORMALLY. I/O WAS H
20.19.29 JOB 1640  +           SYSTEM ABEND CODE=0322-0000 PSW=0000000000000000
20.19.29 JOB 1640  +           EXECUTION MODE=ADVANCED
20.19.29 JOB 1640  +           ENTRY NAME OF THIS PROGRAM IS MAIN     AT LOCATI
20.19.29 JOB 1640  +           VCR0 =DD000096 VCR1 =720A0000 VCR2 =00000000
20.19.29 JOB 1640  +           GR0 =00AFBE28 GR1 =80322000 GR2 =00AFBC28 GR3 =F
20.19.29 JOB 1640  +           GR4 =00FDC138 GR5 =00008410 GR6 =7EF29354 GR7 =0
20.19.29 JOB 1640  +           GR8 =00001CB8 GR9 =00AFBC28 GR10=00008410 GR11=F
20.19.29 JOB 1640  +           GR12=FEF2A598 GR13=000510B8 GR14=FEF2A848 GR15=8
20.19.29 JOB 1640  +           FR0 =0000000000000000       FR2 =BA587073106C33B
20.19.29 JOB 1640  +           FR4 =3E397F3D5C26B843       FR6 =3E72FE79F4B510E
20.19.29 JOB 1640  +           FR8 =406866289D08D460       FR10=000000000000000
20.19.29 JOB 1640  +           FR12=0000000000000000       FR14=000000000000000
20.19.29 JOB 1640  JDJ450I A79999AA GO ABEND S322 U0000
20.19.30 JOB 1640  CD=0322 *** A79999AA (J1640) A79999A : END      TIME=20.19.3
                          :
                          :
```

JWE0019I-U は、 Fortran の発行したエラーメッセージである（JWE で始まるメッセージはすべて Fortran 関連）。エラーの説明から、何らかの原因でプログラムが異常終了したのが解る。これは、 Fortran プログラムの要請ではなく、システムがプログラムの実行に「待った」

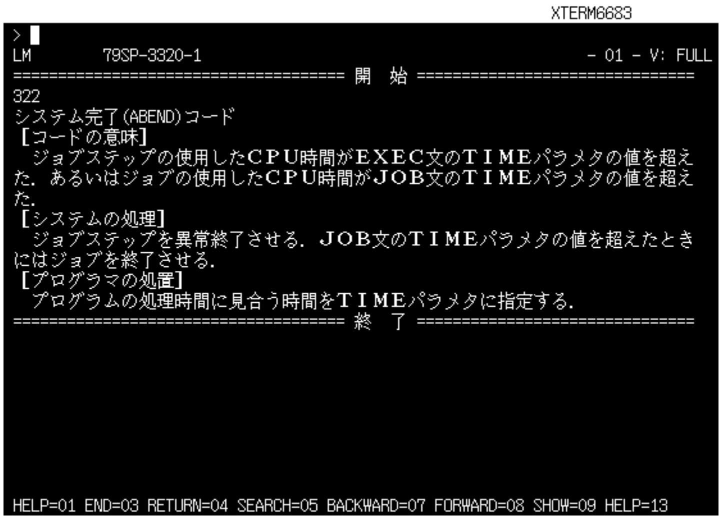
をかけて強制的に終了したことを示す。この場合、システムは「待った」をかけた理由として、３桁のシステム完了コード (ABEND コード) を発行している。

例では SYSTEM ABEND CODE=0322-0000 の‘322’がこれである。最後の CD=0322 のシステム完了コードと同じである。

このシステム完了コードを LM コマンドで検索することにより、異常終了の原因を探る。

```
 READY
LM 322                 <--- LM コマンドの入力
```

LINE モードを省略すると、フルースクリーン環境で、システム完了コードの意味が表示される。



コードの意味により、 CPU 時間オーバーによってプログラムの実行が強制的にキャンセルされたことが解った。対策として、もし TIME パラメータを切っているなら、大きめに設定を変更するか、大きめのジョブクラスで再実行するなどを行なえば良い。

パソコン端末から検索を行なう場合は、必ず LINE オペランドを指定すること。日本語が表示出来ない端末の場合は、英語表示を指定する。

```
 READY
PROFILE TL(E)           <--- 表示を英語に指定
 READY
LM 322 LINE             <--- LM コマンドの入力
322
System ABEND code
<Explanation>
The CPU time used by the job step exceeded the TIME parameter value specified
in the EXEC statement, or the CPU time used by the job exceeded the TIME
parameter value specified in the JOB statement.
<System action>
         :
```

16　ABEND コードの原因と対処

前節の例題でも紹介したシステム完了コード (ABEND コード) で，ここでは，代表的なものについて，原因と対処方法を説明する．

コード	原因	対処方法
0C4	領域外へのアクセスを行なった	DEBUG(SUBCHK) オプションを指定して再翻訳する
0C5	引数の型が一致していない	DEBUG(ARGCHK) オプションを指定して再翻訳する
0C6	配列宣言が誤っている	DEBUG(SUBCHK) オプションを指定して再翻訳する
0C7	領域外へのアクセスを行なった 引数の型が一致していない	DEBUG(SUBCHK,UNDEF,ARGCHK) オプションを指定して再翻訳する
0CF	ゼロ割りが起きている	最適化レベルを下げる プログラムを修正する
222	オペレータによるジョブのキャンセル	センター受付に問い合わせる
322	CPU 時間オーバー	TIME パラメータを大きくとる ジョブクラスを変更する
522	ジョブがシステムの待ち時間を越えた	再度ジョブを投入する
722	出力量が制限値を越えた	ジョブクラスを変更する
804	リージョンサイズ不足	ジョブクラスを変更する
B37 E37	データセットの領域不足	データセットの容量を大きめに設定し再実行する
E22	EXCP 回数が制限値を越えた	ジョブクラスを変更する
913	データセットの利用資格がない	データセットの利用権を確認する

その他，Fortran77 EX の代表的なエラーメッセージもあげる．

メッセージ	種別	現象
JWE0011I	指数オーバーフロー	演算結果の絶対値が 16^{63} を超えた
JWE0012I	指数アンダーフロー	演算結果の絶対値が 16^{-65} より小さくなった
JWE0013I	浮動小数点除算例外	浮動小数点演算でゼロ割りが起きた
JWE0015I	固定小数点オーバーフロー	演算結果の絶対値が $2^{31}-1$ を超えた

参考文献

[1] 「UXP/M for the Working Scientists」佐藤 周行, 九州大学大型計算機センター・ UXP 講習会資料, 1994. UXP の`/usr/local/doc/uxpguide.dvi` に TEX の dvi file あり

[2] 「Unix 環境でスーパーコンピューティングをする人のためのガイド」 佐藤 周行，九州大学大型計算機センター広報, Vol.26, No.4, pp.452–484, 1993.

[3] 「VP2600 のシステム記憶とその使用法」 島崎 眞昭，九州大学大型計算機センター広報, Vol.25, No.3, pp203–209, 1992.

[4] 「OS IV/MSP FORTRAN77 EX 使用手引書」(79SP-5031)，富士通株式会社．
（PLUM コマンドでオンラインマニュアルとして参照可）

[5] 「OS IV/MSP FORTRAN77 EX/VP 使用手引書 V12 用」(79SP-5041)，富士通株式会社

[6] 「OS IV FORTRAN77 EX メッセージ説明書 V12 用」(70SP-5321)，富士通株式会社

[7] 「FORTRAN 新コンパイラの公開について」 竹生 政資, 九州大学大型計算機センター広報, Vol.25, No.1, pp55–60, 1992.

[8] 「利用の手引・バッチジョブ編」 九州大学大型計算機センター・ライブラリ室, 1992.

[9] 「利用の手引・ TSS 編」 九州大学大型計算機センター・ネットワーク室, 1993.

[10] 「OS IV/MSP TSS/E コマンド文法書」(79SP-4091)，富士通株式会社．

[11] 「PFD と PFDE の使用法」 平野 広幸，九州大学大型計算機センター広報, Vol.25, No.2, pp.119–165, 1992.

[12] 「利用の手引・ MSP コマンド編」九州大学大型計算機センター・ライブラリ室, 1993.

[13] 「デバッガ機能の紹介」 渡部 善隆，平尾 耕二, 九州大学大型計算機センター広報, Vol.26, No.1, pp.1–47, 1993.

[14] 「SSL II 使用手引書（科学用サブルーチンライブラリ）」（99SP–0050）, 富士通株式会社．

[15] 「SSL II 拡張機能使用手引書（科学用サブルーチンライブラリ）」（99SP–4070）, 富士通株式会社．

[16] 「ライブラリ・プログラム利用の手引（数値計算編： NUMPAC ）」 名古屋大学大型計算機センター．

[17] 「OS IV デバッガ使用手引書 FORTRAN, C 言語用」(70SP-6430), 富士通株式会社．

[18] 「FACOM OS/MSP チューナ使用手引書 V10L10 用」(79SP-4720)，富士通株式会社.

[19] 「 OS IV/MSP アナライザ使用手引書 FORTRAN 用」(79SP-5090), 富士通株式会社．

[20] 「 OS IV/MSP アナライザ使用手引書 FORTRAN、 VP 用」(79SP-5080), 富士通株式会社．

[21] 「MSP / FORTRAN 利用法 ’94」 肥田木 直子，渡部 善隆，九州大学大型計算機センター広報, Vol.27, No.4, 1994.

【用語集】

ベクトル計算機
Vector Prosessor(VP)．配列データを対象とした DO 文の集合対する繰り返し処理を，演算パイプライン方式により，処理を高速化する機能を持つコンピュータ．

OS
オペレーティングシステム．コンピュータを構成しているハードウェアと，操作する人との間に立ち，人とハードウェアの間での意志のやり取りを仲介する役目を果たすソフトウェア．

MSP
Multidimensional System Products．計算機システムを効率よく運転するための OS（オペレーティングシステム）．システムの監視，ジョブ制御などの機能に優れている．

UXP
UNIX Product. UNIX のオペレーティングシステム．

TSS
Time Sharing System．端末から通信回線を介して多数の利用者が同時に計算機を利用するための対話型の処理形態のこと．

CPU 時間
ジョブを処理するのに消費された CPU(Central Processing Unit: 中央処理装置) の動作時間．ジョブを投入してからの経過時間とは異なり，計算機が本当に仕事をした時間のこと．

ファイルアクセス回数
EXCP(EXecute Channel Program) 回数とも呼ばれる．1 ブロックをバッファに読み込む度に 1 回ずつカウントされる．

リージョンサイズ
プログラムが読み込まれ，実行される領域．

コンパイラ
ユーザが作成したプログラムを計算機が理解可能な言語に翻訳するプログラムのこと．

チューニング
調整．ここでは FORTRAN プログラムを VP で高速実行可能なように修正することと同義．ある程度のチューニングはコンパイラが勝手にやってくれる．

オブジェクトモジュール
コンパイラによって計算機が理解可能な形に翻訳された機械の言葉．最終的な実行可能なプログラムにするためには，リンクと呼ばれる補強作業が必要．

ロードモジュール
実行可能な完全な機械語プログラム．簡単に「モジュール」とも呼ばれる．

バッチ処理
プログラムやデータの処理をひとまとめにしてシステムに依頼する方法．ジョブの進行中，利用者は処理そのものをキャンセルする以外はバッチ処理に介入出来ない．

私用ライブラリ
利用者個人が作成するライブラリのこと．サブルーチン，関数を翻訳，リンクして自分の課題に保存して利用する．

NUMPAC
名古屋大学大型計算機センター提供のサブルーチン・ライブラリ．汎用機用と VP 用をサポート．

SSL II
富士通株式会社提供のサブルーチン・ライブラリ．汎用機用と VP 用をサポート．行列処理（連立一次方程式，固有値問題），数値微積分，FFT などの代表的解法は，SSL II, NUMPAC のどれかを捜せば大体登録されている．

汎用コンピュータ
一般的な用途のために設計されたコンピュータ．様々なアプリケーションソフト（統計解析，線形計画問題，数式処理，図形処理，磁気テープ処理, etc.）が動作する．それに対して，VP で動作するソフトは基本的に FORTRAN で記述されており，科学技術計算に目的が絞られる．さらに，素粒子物理学シミュレーションの計算のみの目的で開発された専用の計算機などもあり，コンピュータにも色々ある．

SSU
System Storage Unit. メモリーとディスクの中間に位置する記憶装置．

アナライザ
FORTRAN プログラムの動作情報を収集分析し，ベクトル計算機での実行性能改善を支援するツール．

VIO/F ファイル
メモリー，SSU に割り当てられた作業用の一時ファイル．