資　料

FORTRAN プログラム動的解析システムの移し換えについて*

藤村直美** 牛島和夫**

Abstract

Easily transferable software is desirable for development and use between various computer systems. However, we have some difficulty transferring software from one computer system to another. It is mainly because of the difference of computer systems as practical environments.

We transferred a FORTRAN dynamic analyzing package, which provides a program execution profile and was first implemented on FACOM 230-45S OS II, to FACOM 230-75 Monitor 7 and HITAC 8800/8700 OS 7. We encountered some problems during our implementation, e. g. file management, Job Control Language and cpu time measurement method. This paper is a report of those problems and how we managed them.

1. はじめに

ある処理系で価値を認められたソフトウェアが別の処理系に困難なく移し換えられて能率良く働くことができれば，ソフトウェアの作成・使用の両面で有意義である．移し換えたいソフトウェアが FORTRAN, ALGOL あるいは COBOL のような標準規格を持つ高級言語で記述されていて，新しい計算機システムにそれらの処理系が用意されていれば移し換えは一応うまくいくと考えられる．またマクロ処理系の上で移し換えを行うものなどがいろいろ工夫されている[1),2)]．

しかしながら問題のソフトウェアの動作環境としての計算機システムを考えると，各々異なった設計思想を持って発展してきており，高級言語の標準規格に陽に現れないような動作環境の相違が問題のソフトウェアの実行ないしは結果に影響を与えることもしばしばである．こういった動作環境に関する情報はマニュアル（利用手引書，計算センターニュースなど）から得られるが，移し換えのためには必ずしも充分でない．あるいはたまたま情報が入手できても各処理系で，同じ概念を異なる用語で表す．あるいはその逆など解釈上の問題もある．これは特に新しい不慣れな計算機システムを使用する時に問題となる．

ここでは FORDAP システムと呼ばれる **Fig. 1** のシステム構成を持つ比較的小規模なシステムの移し換え作業で直面した問題点を中心に述べる．すなわち九

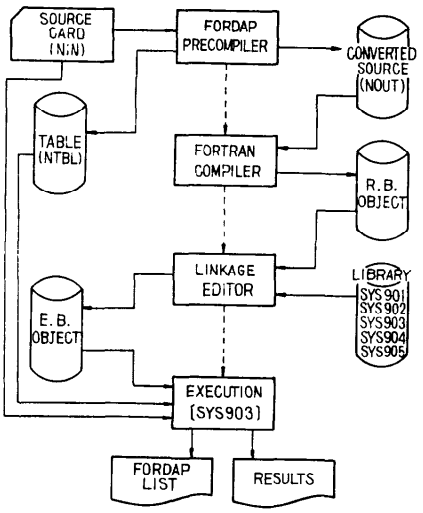

Fig. 1 General flow of FORDAP system

* On transferability of FORTRAN dynamic analyzing system by Naomi FUJIMURA and Kazuo USHIJIMA (Faculty of Engineering, Kyushu University)

** 九州大学工学部情報工学科

州大学情報工学科の FACOM 230-45S OS II (以後 OS II と呼ぶ)において最初設計作成した FORTRAN プログラムの動的解析システムを他の計算機システム，すなわち九州大学大型計算機センターの FACOM 230-75 モニター Ⅶ (以後 M 7 と呼ぶ) と東京大学大型計算機センターの HITAC 8800/8700 OS 7 (以後 OS 7 と呼ぶ) に移し換えを行った．なお M 7 には FORTRAN D (本来 FACOM 230-60 用に作成されたコンパイラ，以後 M 7 D と呼ぶ) と FORTRAN H (以後 M 7 H と呼ぶ) と呼ばれる二つのコンパイラがある．

この動的解析システムの手続きは FORTRAN で記述されているので大部分の移行は比較的容易に行うことができたが，実際の運用に際し処理系特有の OS のもとで，ファイル，ジョブ制御文，時間計測等，動作環境の相違が原因となる幾つかの問題に直面した．ここではこういった問題に対して我々が採った方策とその結果について述べ，ソフトウェア移し換えの一つの資料としたい．

2. FORDAP システム

プログラムは読んで理解し易いことが大切である．その上で効率の良いことが望まれる．Ingalls[3] の報告によると I/O bound でないプログラムの実行時間の大部分はプログラムの原始テキストの数%に集中しているものが多いという．そこでまずプログラムを理解し易いように記述し，問題の数%に工夫をこらせば全体として理解し易く効率の良いプログラムが作成できると考えられる．そういう目的のために Ingalls[3]，Knuth[4] を参考にして FORTRAN プログラムの動的解析システムを作成した[5),6)]．これを Knuth[4] にならって FORDAP システムと呼ぶ．

2.1 機能と設計上の考慮点

FORDAP システムは次のような機能を持っている．

(1) FORTRAN プログラムの各実行文の実行回数を計数する．論理 IF 文ではさらに論理式の値が真となる場合の数を計数する．

(2) プログラムの実行コストを与える．

(3) 上記の情報を理解し易い形で出力する．

さらに最初に OS II で FORDAP システムを作成するにあたり考慮した主な点は次のようなものである．

(1) 他の処理系への移換性を考慮してプログラムの記述を可能な限り FORTRAN JIS 7000 の範囲で行う．

(2) デバッグ支援のため異常終了時にも可能な限り情報を提供する．

(3) 実行コストの算定を容易にする．

実行コストの決定にはプリコンパイラが原始プログラムをかなり詳細に構文解析し，各文の等価的なコストを算定する方法がある[3)]．しかしこの方法はプリコンパイラの負担が増加し，処理時間も増大する．そこで実行コストには実際の cpu 時間をあてることとし，時間計測のオーバヘッドを考慮して計測の単位を各実行文単位でなく各プログラム単位とする．しかしながら cpu 時間の計測は処理系に依存した機能を利用する事になり，ここで移し換えたシステムの実際の運用面で幾つかの問題が生ずることがある (4.2 で詳述)．

2.2 システムの構成

現在 OS II のもとで稼動している FORDAP システムの構成が Fig. 1 である．1 パスの FORDAP プリコンパイラが原始プログラムに 2.1 で述べた機能を果すべく文を追加編集し，その編集された原始プログラムがコンパイル後実行されることになる[5)]．TABLE には収集された情報を最後に原始プログラムに付加してリストを編集するための補助情報を貯える．**Table 1** は LIBRARY 中の SYS 901～SYS 905 の機能を示す．

2.3 移し換えのための条件

Fig. 1 で示されたシステム構成を持つ FORDAP システムを全体としてそのまま他の処理系へ移し換えるには相手の処理系に次のような機能が要求される．

A1: FORDAP プリコンパイラが正しく動作すること．

A2: FORTRAN プログラムでファイルが自由にかつ必要なだけ使用できること．

A3: FORTRAN プログラムによって作成されたファイル(Fig. 1 NOUT に相当)が FORTRAN コンパイラの入力となり得ること．

A4: プログラムの結合・編集の段階で外部記憶 (Fig. 1 LIBRARY に相当) から作成済みの副

Table 1 Contents of LIBRARY

名　前	機　能
SYS 901	時間測定のための初期設定
SYS 902	時間測定のための積算
SYS 903	リスト編集
SYS 904	異常終了の処理
SYS 905	異常終了の処理

プログラムを組み込めること.

A5: ユーザレベルで cpu 時間の計測ができること.

A6: 異常発生時に何らかの形でユーザプログラムによる後処理が可能であること.

3. 移し換えの可能性と実現方法

移し換えの作業は **2.3** で述べた条件 A1～A6 の可能性を調べ実現方法（ジョブ制御文など）を見出す所から始める必要がある（A5 は **4.2** で詳述）.

3.1 プリコンパイラ

ここでは A1 の検討を行う. プリコンパイラは可能な限り FORTRAN JIS 7000 の範囲内で記述されているが, 次に示す点が JIS の規格外である[6].

（1） プリコンパイラは必然的に文字の比較を行うが, JIS 規格では文字の比較が定義されていない.

（2） 入力文でファイルの終端に達した時, 必要な後処理を行うために READ (u, f, END=n) の形の入力文を使用している.

（3） プリコンパイラでは文字の種類を区別する関数が極めて多数回引用されるので, 実行速度向上のためこの部分に限りハードウェアに依存した形とした(実行文8文). そこで移し換えに先立ってこの関数が各処理系で正しく働くことを確認するためのテストジョブを必要とした.

（4） 整数型データの暗黙のビット長に対する配慮のため宣言文の一部に INTEGER *4 を使用している.

3.2 ファイルの仕様の決定

ここでは **3.3** とあわせて FORDAP システムで使用するファイルの仕様を決定するための検討を行う. ファイル編成, ファイル参照番号, レコードサイズなどは各処理系に依存せざるを得ない. ここでは A2 を中心に検討する. 単にファイルといっても各処理系で種々の形態があり, 全てを FORTRAN で使用できるわけではない. FORDAP システムではどの処理系でも使用可能な順編成ファイルを使用している.

各処理系で使用できるファイル参照番号（データセット参照番号, 入出力装置番号などとも呼ばれる）は OS II では 1～50, M 7 と OS 7 では 1～99 であり, これが各処理系で一度に使用できるファイルの個数の上限となる. FORDAP システムで使用するファイル参照番号はユーザが使用するファイル参照番号と重複しないことが望ましい (Fig. 1 の EXECTION の時). プリコンパイラでは一般性のためにファイルの参照は整変数 (NIN, NOUT, NTBL) で行っている. ところがこうすると, M7D でファイル参照番号の実際の値が 1～8 では問題ないが, 9～99 となると少なくとも一度入出力文にファイル参照番号を陽に記述する必要がある, あるいは M7H では 6 個以上のファイルを同時に使用する時はジョブ制御文にその個数を与える必要があるなどの制限があることがわかった. これは通常特定の処理系でのみ仕事をしていると仲々気付かない移し換えに特有の例といってよかろう.

ファイル参照番号はプログラムの実行時に具体的なファイルと結合するために何らかの手続きでファイル定義名に対応付けられる（**Table 2** 参照）. M 7 と OS 7 では同じファイル参照番号に異なったファイル定義名が対応し得る. これは次節で述べるようにジョブ制御文の作成に際して問題となる.

実際にはさらにレコードサイズその他の情報が必要である. これらの情報を OS II ではコンパイラに対する PARA 文か OPTION FILE 文で与え, M 7 と OS 7 ではジョブ制御文の FCB パラメータとして与える. この時 M 7 では主記憶とファイル間のデータ転送に 8 ビットモードと 9 ビットモードがあるのでレコードサイズの計算に注意が必要である.

3.3 FORTRAN コンパイラの入力形成

ここでは A3 について検討する. ここで論じている処理系のコンパイラはいずれもカードリーダの他に種々の形式のファイルから原始プログラムを入力するこ

Table 2 File reference number (FRN) and File definition name

OS / FRN	OS II	M 7			OS 7		
		(1)*	(2)	(3)	(1)	(2)	(3)
5	UIN	F 05.001	READ		FT 05 F 001	SYSIN	SYSMIN
6	LIST	F 06.001	PRINT	SYSPRT	FT 06 F 001	SYSLST	SYSMSG
7	U 07	F 07.001	SYSPCH (PUNCH)		FT 07 F 001		
n	Un	F n .001			FT n F 001		

priority: (1)>(2)>(3), (1)* is not available at M 7 D

とができる．この入力形式は各処理系でまちまちであるが，大別するとシステム入力としての形式（順編成ファイル）とライブラリ入力としての形式に分類される．いずれかの形式を満足するファイルをFORTRANプログラムで作成できれば，そのファイルをコンパイラの入力とすることが原理的には可能である．

3.2 で述べたようにプリコンパイラは順編成ファイルを想定しているので，移換先の処理系においてFORTRAN の原始プログラムからなる順編成ファイルを FORTRAN プログラムで作成し，コンパイラに渡すことができるか否かが移し換えの鍵となる．これを知るためのテストプログラムを各処理系のジョブ制御文も含めて **Fig. 2** に示す(M7H は省略)．Fig.2 は最終的に成功した例であるが，これが得られるまでには数回のジョブを流すことを余儀なくされた．例えばM7D ではこのテストに関する情報が何も入手できなかったので，レコードサイズは 80 バイトを仮定しブロック化率を変えて行った．その結果M7D ではブロッキングを行っていないことが判明した．M7H でのテストはたまたま文献 8) を参照できたので意外に簡単に成功した．OS 7 ではマニュアル[9]の情報を参考にしてファイルの引き渡しは簡単に成功したが，最初

```
#NO
#USER
#WJOB
#FORTRAND
C      COMPILER INPUT FILE TEST
C
       INTEGER CR(40)
       NIN=5
       NOUT=8
  100 READ(NIN,10,END=900) CR
       WRITE(NOUT,10) CR
       GO TO 100
C
  900 ENDFILE NOUT
       STOP
   10 FORMAT(40A2)
       END
#LIEDRUND
***  FORTRAN PROGRAM SOURCE DECK  ***
#FD  FU8,FILE=(TEMP,A),UNIT=DPO,VOL=WORK,DISP=PASS,RCDSIZE=80,
          BLKSIZE=80,EOFNO=2,BUFL=80
#FORTRAND    STEP=1
#FD  FORTRAND1=SOURCE,FILE=(TEMP,A)
#LIEDRUND    STEP=1
***  DATA DECK  ***
#JEND
```

(a) M7D

```
//TEST:    JOB
//FORTCOBOL:   DTF FN=&OUT,DISP=(OLD,DELETE,DELETE),FCB=(RECL=80,BLKSIZE=1200,RECFM=FB),
          DISP=(NEW,PASS,PASS),FORG=S
//FORTCG   (SOURCE,NOMAP,DIAG=1,NOESD,NAME=MAIN)
 same test program shown in Fig.2 (a)
/*
***  FORTRAN PROGRAM SOURCE DECK  ***
//STEP
//SYSIN:   DTF FN=&OUT,DISP=(OLD,DELETE,DELETE)
//SYSOBJ:  DTF FN=&FORT,SYSOBJ,FORG=PP,DISP=(NEW,PASS,PASS)
//      FORTC  (SOURCE,NOMAP,DIAG=0,NOESD,OPT=0)
//STEP
//    RUN  FN=&FORT,SYSOBJ
***  DATA DECK  ***
//END
```

(b) OST

Fig. 2 Test jobs for passing a created file

Table 3 File constants of FORDAP systems

OS	FILE	FRN	RCDS	BLKS	BF	BN	RECFM	CWL
OS II	NIN	41	80	1040	13	2	F	0
	NOUT	42	80	1040	13	2	F	0
	NTBL	43	4	1024	256	2	F	2
M7D	NIN	5→7	80	80	1	2	F	0
	NOUT	8	80	80	1	2	F	0
	NTBL	9	9	360	1	2	F	4
M7H	NIN	5→7	80	1200	15	2	F	0
	NOUT	8	80	1200	15	2	F	0
	NTBL	9	9	360	40	2	F	4
OS 7	NIN	41	80	80	1	*	F	0
	NOUT	42	80	1200	15	*	FB	0
	NTBL	43	12	1024	85	*	VB	4

FRN: file reference number, RCDS: record size,
BLKS: block size, BF: blocking factor,
BN: buffer number, RECFM: record format,
CWL: control word length, *: unspecified,
F: fixed length (blocked) record,
FB: fixed length blocked record,
VB: variable length blocked recood.

ジョブステップの概念が他二者と異なることに気付かず，//RUN の直前に //STEP を挿入しなかったために SYSMIN として作成されるデータファイルより，先に陽に定義されている SYSIN が優先され実行段階で異常終了した (Table 2 参照)．移し換えに先立って情報を得るために行うこのような簡単なテストプログラムの移し換えでもかなり手間取ることが多い．

このようにして各処理系における FORDAP システムのファイルの仕様は **Table 3** に示す値に決定することができた．

3.4 プログラムの結合・編集

ここでは A4 について検討する．FORDAP システムでは時間計測その他の機能を持った副プログラムを用意する (Table 1 参照)．プログラムの結合・編集段階でこれらの副プログラムを組み込む機能が要求される (Fig. 1 参照)．各処理系によってこの機能の具体的な実現方法が異なる．各処理系の概略は次の通り，ただし実現のために必要な制御文は **4.1** で述べる．

(1) OS II の場合：

オブジェクトプログラムファイルはコンパイラが出力したままの形式とユーティリティプログラム EDITSL により作成される自動組込みに適した形式がある．前者では陽に組込みを指示する必要がある．ここでは後者の自動組込み機能を利用する．

(2) M7の場合:

OS II とほぼ同様であるが，ファイル定義名を陽に指定して組み込む方法を採る．

(3) OS 7 の場合:

オブジェクトプログラムファイルの形式は1通りであるが，結合・編集の方法にはリンケージエディタへの制御文によって陽に組み込むプライマリ入力による方法，自動組込み機能を使うライブラリ入力による方法があり，さらに実行時に結合するダイナミックリンクによる方法がある．ここでは最初の方法を利用する．

こういったプログラムの結合・編集機能は OS の基本設計あるいは計算センターの運営方針に直接，間接に影響されると考えられ，入念に検討する必要がある．

3.5 異常終了の処置

実行中のプログラムにおいて何らかの異常が発生した場合の取扱いは各処理系によって異なる（A6 の検討）．

(1) OS II の場合:

OS II FORTRAN S では組込み関数，外部関数等で発生するエラーはそのルーチン内で閉じているので対処できない．OS II では実行時間の予定超過を含む16 通りのエラー[11]発生時に限ってユーザが対処できる．その方法は **Fig. 4**(a)(次頁参照) に示すようにリンケージエディタへの制御文／UBA によって行う．すなわち前述の 16 項目のエラー発生時には制御が副プログラム SYS 904 (**Fig. 3**(a)参照) に渡され，SYS 904 が二つの副プログラム SYS 903 と SYS 905 を使ってリストの編集，エラーコードの出力を行う．

(2) M7D の場合:

ユーザレベルでは異常終了に対処する方法がない．

(3) M7H の場合:

実行時に発生するエラーは全て FORTRAN H のエラーモニターがエラー制御表に従って管理する．エラー制御表は変更可能でプリコンパイラが原始プログラムの最初の実行文の直前に CALL SYS 904 (Fig. 3 (b) 参照) を挿入することにより行う．その結果エラーが発生した時には利用者訂正ルーチンとして登録した異常処理ルーチン SYS 905 に制御が渡る．この時エラーモニターは必要な情報を引数の形で SYS 905 (引数無し) に渡そうとするが，引数の個数はエラーの種類で異なり不都合が生じる．これはあらかじめ SYS 905 をコンパイルする時に引数の個数のチェックをしないように指定 (NOARGCHK) することで回避できた．なおエラー制御表を変更できない一部のエラーには対処できない．

```
SYS904   ELMNT  LIST,CROSS
SYS904   SECT   1
         SBOUND 1
SYS904   DA     2
*        CALL SYS903
             B      SYS903,,,1
             DC     S'0'
*        ERROR CODE SET
             LD     SYS904,,4,1
             STD    ID,,4,0
*        CALL SYS905(ID)
             B      SYS905,,,1
             DC     S'1'
             DC     A'ID'
+            RETURN AOS
*
DATA     SECT   0
         SBOUND 1
ID       DA     2
*
         GLOBAL SYS904
         EXTRN  SYS903,SYS905
*
         END
```

(a) OSII

```
SUBROUTINE SYS904
EXTERNAL SYS905
CALL ERRSET(600,2,0,2,SYS905,612)
CALL ERRSET(614,2,0,2,SYS905,619)
CALL ERRSET(621,2,0,2,SYS905,629)
CALL ERRSET(631,2,0,2,SYS905,658)
CALL ERRSET(660,2,0,2,SYS905,752)
RETURN
END
```

(b) MTH

```
SUBROUTINE SYS904
EXTERNAL SYS905
CALL ERRSET(201,2,2,2,SYS905,211,2)
CALL ERRSET(214,2,2,2,SYS905,228,2)
CALL ERRSET(231,2,2,2,SYS905,231,2)
CALL ERRSET(234,2,2,2,SYS905,332,2)
RETURN
END
```

(c) OS7

Fig. 3 Error routine SYS 904

(4) OS 7 の場合:

基本的な概念は M7H の場合とほぼ同じであるが，SYS 905 に制御が渡る時の引数の個数をチェックしないようにする方法が不明であったために実用上の効果が制限される．なお各エラーに対してエラー制御表の変更の可否に関する情報が不明であったので試行錯誤のためのテストジョブを必要とした．Fig. 3(c) に OS 7 における SYS 904 を示す．

4. 各処理系における FORDAP システムの使用

前章で FORDAP システムの移し換えに要求される条件 A1～A4, A6 について検討した．そして M7D と OS7 の異常終了の処置に関する問題を除けば何とか実現の方法があることがわかった．ここでは FORDAP システムの使用に関連してジョブ制御文と時間計測について述べる．

4.1 ジョブ制御文

FORDAP システムを使用するために必要なジョブ

制御文を **Fig. 4** に示す. Fig. 4(a) は OS II の場合で全て基本制御文で記述している. 常にこれだけの制御文を用意するのは不便なので Fig. 4(b) のようにマクロ化を行っている. Fig. 4(c) が M7H におけるジョブ制御文で基本ジョブ制御文とマクロ文が混在している. 九大センターでは一般ユーザがジョブ制御文のマクロ化を行うことを認めていない*. Fig. 4(d) が OS7 におけるジョブ制御文で全てスタティクリンクを行っている. OS 7 では一般ユーザでもジョブ制御文のマクロ化が可能であり, 検討中である. こういったジョブ制御文のマクロ化は OS の機能と計算センターの運営方針に大きく依存する.

Fig. 4 (a), (c), (d) から各処理系における OS の基本設計の違いの一部が伺われる.

(1) OS II と OS 7 では Fig. 4(a), (d) の(1)で示されるようなジョブ制御文で OS によって作成される寸借ファイルを後続のジョブステップ (各処理系でジョブステップの概念が異なる) に渡すことができるが, M 7 ではできない. そのため M 7 ではファイル定義名 READ として作成される寸借ファイルを SYSPCH へコピーし, そのファイルを後続のジョブステップに渡さざるを得ない. その結果 M7 に移換されたシステム構成は Fig. 1 に示したものとは若干異なることになった.

(2) ファイルを後続のジョブステップに渡す時, OS II では Fig. 4 (a) の (2)のようにファイル定義名で渡す. 一方, M 7 と OS 7 では Fig. 4 (c), (d) の(2), (5)で示されるようにファイル名で渡す. そのため M7 と OS7 では後続のジョブステップでも各々に対応したジョブ制御文が必要となる.

```
        *  JOB FORDAP
        *  EX  FORDAP,NOMNT,PRTY=3,FILE=FUJI.ELIB,FORDAP,VOL=000001
 (1)----*  FD  U41=*,VOL=000000,DISP=CONT
        *** FORTRAN PROGRAM SOURCE DECK  ***
        *  FD  U42=DA,VOL=000001,TRK=(20,20,RLSE),DISP=(CONT,CLOSE)
        *  FD  U43=DA,VOL=000001,TRK=(20,20,RLSE),DISP=(CONT,CLOSE)
        *  FD  LIST=DA,VOL=000000,TRK=(20,20,RLSE),SOUT=A
        *  PARA  MSGLEVEL=8
        *  EX  FORTRAN,NOMNT,PRTY=3,SIZE=32
        *  FD  LIST=DUMMY
        *  FD  RLIB=DA,VOL=000000,TRK=(20,20,RLSE),DISP=(CONT,CLOSE)
 (2)----*  SW  SLIB=U42
        *  PARA  NOMAP,LINE=0,NOSEQ,ICF
        *  EX  LIED,NOMNT,PRTY=3,COND=20
        *  FD  LIST=DA,VOL=000000,TRK=(20,20,RLSE),SOUT=A
        *  FD  ELIB=DA,VOL=000001,TRK=(20,20,RLSE),DISP=(CONT,CLOSE)
        *  FD  ALIB=DA,VOL=000000,FILE=F.ALIB
 (3)----*  FD  U03=DA,VOL=000001,FILE=FUJI.ALIB,FORDAP
        *  FD  COIN=*
 (4)----/  NAME  RUN,SEP,CALL
        /  UBA  SYS90,MASK=X'FFFF'
        /  SEGMENT  AA
        /  SELECT  RLIB
        /  FIN
        *  EX  RUN,FILE=ELIB,NOMNT,PRTY=1,TIME=5,COND=20
        *  FD  LIST=DA,VOL=000000,TRK=(80,40,RLSE),SOUT=A,LIMIT=100
        *  FD  U16=DA,VOL=000001,TRK=(80,40,RLSE),SOUT=A
        *  PARA  MSGLEVEL=8
        *  FD  UIN=*
        ***  DATA DECK  ***
        *  JEND
```

(a) OS II

```
        *  JOB FORDAP
        *  FORDAP          *
        ***  FORTRAN PROGRAM SOURCE DECK  ***
        */
        ***  DATA DECK  ***
        *  JEND
```

(b) Macro JCL for OSII

```
     #NO
     #USER
     #JOB
(1)--- #RUN      EBNAME=FORDAP,ESFLNAME=F1400.ELIB
     ***  FORTRAN PROGRAM SOURCE DECK  ***
     #FD  SYSPCH,FILE=(TEMP,A),UNIT=DPO,VOL=WORK,DISP=PASS,RCDSIZE=80,      -
            BLKSIZE=1200,BUFNO=2
     #FD  F08,FILE=(TEMP,B),UNIT=DPO,VOL=WORK,DISP=PASS,RCDSIZE=80,        -
            BLKSIZE=1200,BUFNO=2
     #FD  F09,FILE=(TEMP,C),UNIT=DPO,VOL=WORK,DISP=PASS,RCDSIZE=9,         -
            BLKSIZE=360,BUFNO=2
     #FORTRANH        NOLIST
(2)--- #FD  FORTRANH=SOURCE,FILE=(TEMP,B)
     #LIED      NOLIST
     NAME   EXWTPRGM,ENTRY=ELM(MAIN)
     CALL       SYSLIB.FORTLIBH.F.SSLH.F.LIB
(4)--- SELECT   RELBIN,RLIB
     FIN
(3)--- #PROFILE    RLIB,F1400.ELIBH
     #RUN      STEP=1
     *** DATA DECK  ***
(5)--- #FD  F08,FILE=(TEMP,A)
     #FD  F09,FILE=(TEMP,C)
     #JEND
```

(c) MTH

```
     //FORDAP:  JOB
(1)--- //FT41F001: DTF  *,FN=&NIN,DISP=(NEW,PASS,PASS)
     *** FORTRAN PROGRAM SOURCE DECK  ***
     //FT42F001: DTF  FN=&NOUT,FCB=(RECL=80,BLKSIZE=1200,RECFM=FB),FORG=S,  -
            DISP=(NEW,PASS,PASS)
     //FT43F001: DTF  FN=&NTBL,FCB=(RECL=12,BLKSIZE=1024,RECFM=VB),FORG=S,  -
            DISP=(NEW,PASS,PASS)
     //         RUN     FN=(RLIB,FORDAP)
(2)--- //SYSIN:   DTF  FN=&NOUT,DISP=(OLD,DELETE,DELETE)
     //SYSOBJ: DTF  FN=&FORT.SYSOBJ,DISP=(NEW,PASS,PASS),FO=PP
     //         FORTC  UNUSOURCE,NOMAP,DIAG=0,NOESD,OPT=2)
     //STEP
     //SYSOBJ: DTF  FN=&FORT.SYSOBJ,DISP=(OLD,DELETE,DELETE)
     //SYSOPM: DTF  FN=&FORT.SYSOPM,DISP=(NEW,DELETE,DELETE),FO=PP
(3)--- //RLIB:   DTF  FN=RLIB
     //SYSLNK1:  DTF  FN=&SYS.SYS3.SYSLNKF
     //         LNK     OPTION=(AUTO)
       INCLUDE  SYSOBJ
(4)---  INCLUDE  RLIB
       END
(5)--- //FT41F001: DTF  FN=&NIN,DISP=(OLD,DELETE,DELETE)
     //FT43F001: DTF  FN=&NTBL,DISP=(OLD,DELETE,DELETE)
     //         RUN     FN=&FORT.SYSOPM
     ***  DATA DECK  ***
     //END
```

(d) OS7

Fig. 4 Examples of JCL for FORDAP system

* FORDAP システムが九大センターのサービスプログラムとして登録されることになり, センターの手でマクロ化が行われている.

(3) **3.4** で述べたように各処理系でプログラムの結合編集に異なった方法を採用した．OS II では Fig. 4(a) の(4)で示す CALL 指定で同(3)で定義するライブラリ（ファイル定義名 U03）から自動組込みを行う．一方，M7 と OS7 では Fig. 4(c), (d) の(3)で定義するライブラリ（ファイル定義名 RLIB）は同(4)に示すように陽に組込みを指示する必要がある．

4.2 時間計測

ここでは A5 とも関連して FORDAP システム使用時の cpu 時間の計測に関する問題について述べる．例として次の5種のプログラムを各処理系の FORDAP システムを利用して cpu 時間を計測した結果を **Table 4** に示す．

例1） FORDAP プリコンパイラ自身（文献6）．副プログラム 10 個のうち文字の種類を区別する関数（**3.1** 参照）が 5,685 回引用されている．

例2） 三項方程式の解法プログラム(文献14, Fig. 1)．副プログラムが7個で各々1～6回引用されている．

例3） 三項方程式の解法プログラム(文献14, Fig. 3)．FORDAP システムから得られた情報を参考にして例2のプログラムを改善したもの．

例4） RKG 法により常微分方程式を解くプログラム．副プログラム2個のうち一つは微分方程式（$y'=1$）の右辺を計算するもので，4,996 回引用されている．

例5） 上のプログラムで微分方程式の右辺の計算を副プログラムの代りに文関数にしたもの．

Table 4 で F は FORDAP システムによる所要時間，R は FORDAP システム無しの実所要時間であり，各々3回計測した平均値である．F 時間は全体の所要時間に対する各プログラム単位の寄与率の目安となるべきもので，R 時間をできるだけ忠実に反映していることが望ましい．実際には FORDAP システムを利用すると計測のオーバヘッドのため F 時間は R 時間より大きくなる．計測のオーバヘッドは各種の実行回数を計数するためのものと，cpu 時間を計測するためのものに分けられる．cpu 時間の計測は各プログラム単位が引用されるごとに，その入口で初期設定（SYS 901, Table 1 参照）し，出口で積算（SYS 902, Table 1 参照）する方法を採っている．従って副プログラムを多数回引用するようなプログラムでは時間計測に要する時間が無視できなくなる可能性がある．

Table 4 Comparison between CPU times with (F) and without (R) FORDAP system (unit: msec)

OS / example	OS II (CLOCKM)		OS II'	M7D		M7H		OS7	
	F	*R*	*F*	*F*	*R*	*F*	*R*	*F*	*R*
1	15959	12181	14661	3784	4700	5436	6000	10402	7738
2	4451	2881	4423	160	126	188	159	385	263
3	2626	2062	2624	106	94	143	135	251	213
4	2024	781	1238	273	78	11	129	1899	164
5	694	672	694	72	72	48	48	49	54

OS II': with a timer codedin an assembly language

OS II では最初 FORTRAN S で利用可能な時間計測ルーチン CLOCKM（計測単位 1 msec）を利用した（計測所要時間 239.3 μsec）．Table 4 OS II の例 2, 3, 5 では各プログラム単位とも引用回数が少ないため F 時間は R 時間を充分忠実に反映していると考えられる．一方例4は常微分方程式の右辺を計算する関数副プログラム（実行文は F=1.0 と RETURN のみ）の時間計測のオーバヘッド（239.3 μsec×4996＝1196 msec）が全体の F 時間（2,024 msec）に対して無視できないと考えられた．実際同じ右辺計算を文関数化した例5では F 時間（694 msec）と R 時間（672 msec）はほぼ等しい．そこでアセンブラ語で，より高速の時間計測ルーチン（計測単位 16 μsec，計測所要時間 78.7 μsec）を作成した．これにより例4で著しい改善が見られた（2,024→1,238 msec, Table 4 OS II, OS II′ 参照）．

OS7 でも FORTRAN で利用可能な時間計測ルーチン CLOCK（最小計測単位 0.1 msec）を利用した（計測所要時間 354 μsec）．例1～3はほぼ問題ないと考えられる．例4は F 時間（1,899 msec）の大部分（354 μsec×4996＝1,768 msec）が右辺計算の副プログラムの時間計測に費やされていることがわかる．これでは我々の目的に適っているとは言い難い．OS7 に不慣れなためにアセンブラ語による時間計測ルーチンの作成は行えなかったが，OS II と同様に改善の余地は充分にあると考えている．例5では F 時間（49 msec）より R 時間（54 msec）が大きくなっている．これは OS 7 特有の仮想記憶，高速バッファメモリその他の影響によるものと考えられる．

一方，M7 でも FORTRAN で標準的に利用可能な CLOCKM（計測単位 1 msec）を利用したが，1 msec という計測単位はこの程度高速の計算機としては大きすぎると考えられる．例2, 3, 5では問題ないが，例1, 4のように特定の副プログラムを多数回引用する

例では計測された値がおかしくなる ($F<R$, Table 4, M7D, M7H 参照). これは例4のような短時間に集中して CLOCKM を引用することになるプログラム程著しい. 同じものを文関数にした例5では右辺計算の関数の引用がなくなって, 納得の行く計測値 ($F=R$) が得られている.

5. おわりに

三つの OS のもとで四つの FORDAP システムを作成したことになるが, 各システムごとに特色がでている. まず使い易さの点からジョブ制御文のマクロ化は必須である. 次に異常終了時の処理はプログラムのデバッグ支援として有用であるが, 無限ループに陥るなどした時の実行時間の予定超過に対して有効な OS II と FORTRAN のライブラリレベルで発生するエラーをほとんどがカバーしている M7H が強力なデバッグの手段となり得る. プログラムの実行コストとして実際の cpu 時間を採用したことは OS II では成功したと考えられる. 一方 M7 と OS7 では **4.2** で述べたような問題が残されているが, 一般のユーザがシステム提供の時間計測ルーチンに立ち入るのは困難であると考えられる. FORDAP システムから得られた情報をもとにプログラムの改善を行った例については Table 4 例 2, 3 及び文献 12), 13), 14) を参照されたい.

M7 及び OS7 への移し換えに際して必要な情報は文献 7), 8) 及び 9), 10) によった. 実際の作業には各処理系とも約1人月を要したが, 大部分は各処理系の情報の収集と確認に費やされた.

Fig. 1 に示したシステム構成を持つ FORDAP システムはごく普通のファイルの使い方をしており, ソフトウェアの形態としてはごく一般的なものと考えられる. それにもかかわらず移し換えは必ずしもスムーズにはいかなかった. これは計算機システム間でのソフトウェアの移し換えという観点にたった情報の整備が不充分であることにも原因があると考えられる. 実際 M7 ではファイルの仕様を決定するための情報入手に多数のテストジョブを必要とした. もう少し詳細な情報があれば移し換えの作業期間の短縮, ないしはこれからの改善も可能であろうと考えている.

ここではこういったソフトウェアの移し換えに際して考慮すべきいくつかのチェックポイントを示し, 同時にそれに関連した情報を提供することができたと考えている. なお本研究の一部は文部省科学費補助金によった.

参考文献

1) 和田英一: ソフトウェア工学とプログラム言語, 情報処理, Vol. 16, No. 10, pp. 848～855 (1975).
2) 松下 温, 山崎晴明, 丹下栄二: プログラム・トランスファラビリティ, 情報処理, Vol. 16, No. 10, pp. 871～879 (1975).
3) D. Ingalls: The execution time profile as a programming tool, R. Rustin ed., Design and Optimization of Compilers, Prentice-Hall, pp. 107～128 (1972).
4) D. E. Knuth: An empirical study of FORTRAN programs, Software, Vol. 1, pp. 105～133 (1971).
5) 藤村, 牛島: プログラムの実行解析システムの作成と使用について, 九大工学集報, Vol. 48, No. 4, pp. 95～100 (1075).
6) 藤村, 牛島: FORTRAN プログラムの動的解析システムとプリコンパイラ, 情報処理, Vol. 17, No. 6, pp. 547～550 (1976).
7) 富士通: FACOM 230 M-V FORTRAN 解説編 (II), EX-061-3-5.
8) 富士通: FORTRAN H 使用手引書.
9) 日立: HITAC 8700 FORTRAN プログラマズガイド, 8700-3-020.
10) 佐藤泰夫: FORTRAN 文法とプログラミング, 東大出版会 (1974).
11) 富士通: FACOM 230 OS II LIED 文法編, 46 SP 1812-4, pp. 341～342.
12) 牛島, 藤村: 計算の手間の評価とプログラムの動的解析システム, 京大数研講究録 250, 計算の手間とデータ構造, pp. 163～180 (1975).
13) 牛島, 昭和 50 年度電気四学会九州支部連大論文集 442.
14) 牛島: 第 16 回情報処理学会大会予稿集 314 (1975).

(昭和51年2月26日受付)
(昭和51年3月29日再受付)