C 2　利用者の立場からのコンパイラテストプログラム

－特に式の評価順序の検査を中心として－

牛島　和夫　（九大工）

1。まえがき

コンパイラのテストの対象となるのは，a）コンパイラの正しさ，b）コンパイラの効率（コンパイル速度，目的プログラムの速度），c）エラーメッセージ，デバッグ用の利便などがある．さらにテストをする主体と目的によって次のような分類もできる．

1）コンパイラライタが仕上りのテストとして

2）コンパイラ提供者（例えばメーカ）が出荷試験として

3）ユーザ側のサービス部門（例えば計算センター）が受入れ検査として

4）個々のユーザがコンパイラ言語を使って問題の解決を行う際に，プログラムマニュアルからだけでは不明かつあいまいなプログラム（あるいは計算機）の動作を知るために

コンパイラの供給者側にたてば，a），b），c）それぞれの設計基準が満たされていることがテストの合格といえよう．一方ユーザ側にたった場合，3）は，別の処理系で実際にrunしたことのあるプログラム群をかなり多数かけてみることで目的を達しているようである．他方4）は，計算機で解決しようとする問題の性質によって，利用者毎に千差万別なテストプログラムが考えられるし，実際に個々のユーザは，その必要に応じてさまざまなテストプログラムを作りその結果から，提供された個々のコンパイラについて情報を得てその利用者の目的とするプログラムの作成に役立てているものと思われる．しかしここで用いられたテストプログラムは，その利用者の目的の補助として作られたものだから，必要な情報が得られゝば，次々と使いすてられて，日の目を見ないまゝに終っているであろう．これらのプログラムのうち一般性のあるものはむしろ集大成して多くの人が同じようなテストをするという無駄を省き，これらのテストを3）あるいは，2）のテストにも利用することが考えられよう．そこでこの論文では，4）のテストの立場から，FORTRAN 語について，メーカの発行するFORTRANマニュアルで言及していない事項* のテストすなわち，①式の評価順序を知るためのテストプログラム，② 丸め誤差の取扱い方法を知るプログラム，③最適化手法がどのように使われているかを知るプログラム…など具体的なプログラムとそれを，C，D，H，N，Tの処理系にかけて得た結果を示す．4）の立場はa）の裏返しで，コンパイラは正しく動作することを前提にして，正しいとは何かを知ることが目的であることもいえる．

＊あるいは索引項目の不備などでマニュアルに書いてあっても探しだすのが困難な事項．これらはむしろ機種毎（implementer毎）にちがう可能性が大きくプログラムの互換性に大きく影響する．[2)]

2. 式の評価順序の検査

2．1　算術式の評価順序の検査

2．1．1　算術式の検査点

FORTRANでは，式の構成順序は原則としてimplementerに任されている．(JISP.19)[1]ところが式の構成順序が原因で処理系毎にちがった結果をひきおこすことがあり，プログラムの実行上の互換性に影響をもつ．すなわち

1）1次子として式に含まれている関数が2つ以上ある場合の評価順序のちがいによって，式の値にちがいをもたらしたり，プログラム全体の結果に影響を与えたりする場合（副作用の問題），

2）浮動小数点演算において，評価順序のちがいから中間結果として格納される要素のちがいによって丸め誤差の発生点のちがいから式の結果に影響を与える場合

一方，多くのFORTRANマニュアルは多少の表現の差はあるが算術式の評価順序について ⅰ）カッコの内部，ⅱ）関数，ⅲ）ベキ乗，ⅳ）乗除算，ⅴ）加減算の順となり，同じ順位の演算子があれば左から右へ計算すると述べている．これは文献2の視点からは不十分である．算術式の構成順序（利用者の立場からみれば評価順序）を知る方法としては，処理系製作者からフローチャートを示してもらう．あるいはアセンブリリストをとるような方法もある．しかし前者は一般に非公開であるし，後者は一般利用者にとって解読は無理であることとアセンブリリストのとれない処理系もある．そこでFORTRANでテストプログラムを作って その計算結果から処理系毎に式の評価順序を知ることにした．テストの関心は主として以下のような点である．

1）式の中に1次子として関数があらわれた場合，同一式内に同じ関数が同じ引数の並びで2個以上ある場合，別の関数が2個以上ある場合の評価順序．

2）目的プログラムの最適化，式の簡単化との関係．

3）単項演算子，二項演算子，多項演算子の取扱い（結合法則をみたす＋×をとくに多項演算子的に取扱うことがありうる）．

4）カッコの有無との関係．

2．1．2　検査プログラムの概要と結果の解釈

JISでは同一式中の副作用を禁じているが，現存する処理系では，副作用のあるプログラムをかけてもチェックされないので，テストにはこれを積極的に逆用する．プログラムを図1に示す（ベキ乗を含むものは省略した）． このプログラムで各代入文VAL(∘)＝×××に先だって共通変数XおよびJの内容を初期化するのがサブルチンRINIT(I)の役割である．図1では紙数の都合でVAL(4)～(49)の直前のCALL　RINIT(I)を省略しているので実行の際は補わねばならない．関数F(I)，G(I)は添字付変数×(I)を関数値とする． これらの関数を引用すると共通変数JとXの値が変化するのでたとえばI＝1のとき

```
CARD-LIST      002635    7001B00667      45/10/22          KYUDAI-CENTER                PAGE-0001

....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8

      SUBROUTINE RARITH
      COMMON/AEXP/ X(1000),J
      DIMENSION  VAL(50)
      DO 1 I = 1, 3
      J = I
      CALL RINIT(I)
      VAL( 1) =   F(J) + X(J) + X(J)
      CALL RINIT(I)
      VAL( 2) =  (F(J) + X(J))+ X(J)
      CALL RINIT(I)
      VAL( 3) =   F(J) +(X(J) + X(J))
      VAL( 4) =   X(J) + F(J) + X(J)
      VAL( 5) =  (X(J) + F(J))+ X(J)
      VAL( 6) =   X(J) +(F(J) + X(J))
      VAL( 7) =   X(J) + X(J) + F(J)
      VAL( 8) =  (X(J) + X(J))+ F(J)
      VAL( 9) =   X(J) +(X(J) + F(J))
      VAL(10) =   F(J) + G(J) + X(J)
      VAL(11) =  (F(J) + G(J))+ X(J)
      VAL(12) =   F(J) +(G(J) + X(J))
      VAL(13) =   F(J) + X(J) + G(J)
      VAL(14) =  (F(J) + X(J))+ G(J)
      VAL(15) =   F(J) +(X(J) + G(J))
      VAL(16) =   X(J) + F(J) + G(J)
      VAL(17) =  (X(J) + F(J))+ G(J)
      VAL(18) =   X(J) +(F(J) + G(J))
      VAL(19) =   F(J) + F(J) + X(J)
      VAL(20) =   F(J) + X(J) + F(J)
      VAL(21) =   X(J) + F(J) + F(J)
      VAL(22) =  -F(J) - X(J) - X(J)
      VAL(23) =  -X(J) - F(J) - X(J)
      VAL(24) =  -X(J) - X(J) - F(J)
      VAL(25) =  -F(J) - G(J) - X(J)
      VAL(26) =  -F(J) - X(J) - G(J)
      VAL(27) =  -X(J) - F(J) - G(J)
      VAL(28) =   F(J) + G(J) - X(J)
      VAL(29) =   F(J) + X(J) - G(J)
      VAL(30) =   X(J) + F(J) - G(J)
      VAL(31) =   F(J) - G(J) + X(J)
      VAL(32) =   F(J) - X(J) + G(J)
      VAL(33) =   X(J) - F(J) + G(J)
      VAL(34) =   F(J) * G(J) + X(J)
      VAL(35) =   F(J) * X(J) + G(J)
      VAL(36) =   X(J) * F(J) + G(J)
      VAL(37) =   F(J) + G(J) * X(J)
      VAL(38) =   F(J) + X(J) * G(J)
      VAL(39) =   X(J) + F(J) * G(J)
      VAL(40) =   F(J) * G(J) / X(J)
      VAL(41) =   F(J) * X(J) / G(J)
      VAL(42) =   X(J) * F(J) / G(J)
      VAL(43) =   F(J) / G(J) * X(J)
      VAL(44) =   F(J) / X(J) * G(J)
      VAL(45) =   X(J) / F(J) * G(J)
      VAL(46) =   F(J)*X(J)+G(J)*X(J)
      VAL(47) =   F(J)*X(J)+X(J)*G(J)
      VAL(48) =   F(J)*X(J)+X(J)*X(J)
      VAL(49) =   F(J)*X(J)+F(J)*G(J)
      WRITE(6,100) I,(K,VAL(K),K=1,49)
  100 FORMAT(1H1/10X,3HI =,I2///(I10,F15.5))
    1 CONTINUE
      RETURN
      END
      SUBROUTINE RINIT(I)
      COMMON/AEXP/ X(1000),J
      DO 1 K = 1,1000
    1 X (K) = K
      J = I
      RETURN
      END
      FUNCTION  F(I)
      COMMON/AEXP/ X(1000),J
      F   = X (I)
      J = 2* I
      X (J) = X (J) +1.0
      RETURN
      END
      FUNCTION G(I)
      COMMON/AEXP/ X(1000),J
      G = X (I)
      J = 3* I
      X (J) = X (J) +1.0
      RETURN
      END

....*....1....*....2....*....3....*....4....*....5....*....6....*....7....*....8
```

図1　プログラム1

No.	arithmetic expression	evaluation sequence 123	132	213	231	312	321
1	f + x + x	7	7	5	3	5	3
2	(f + x)+ x	7	7	5	3	5	3
3	f +(x + x)	7	7	5	3	5	3
4	x + f + x	5	3	7	7	3	5
5	(x + f)+ x	5	3	7	7	3	5
6	x +(f + x)	5	3	7	7	3	5
7	x + x + f	3	5	3	5	7	7
8	(x + x)+ f	3	5	3	5	7	7
9	x +(x + f)	3	5	3	5	7	7
10	f + g + x	11	7	12	9	5	6
11	(f + g)+ x	11	7	12	9	5	6
12	f +(g + x)	11	7	12	9	5	6
13	f + x + g	7	11	5	6	12	9
14	(f + x)+ g	7	11	5	6	12	9
15	f +(x + g)	7	11	5	6	12	9
16	x + f + g	5	6	7	11	9	12
17	(x + f)+ g	5	6	7	11	9	12
18	x +(f + g)	5	6	7	11	9	12
19	f + f + x	9	7	9	7	5	5
20	f + x + f	7	9	5	5	9	7
21	x + f + f	5	5	7	9	7	9
22	-f - x - x	-7	-7	-5	-3	-5	-3
23	-x - f - x	-5	-3	-7	-7	-3	-5
24	-x - x - f	-3	-5	-3	-5	-7	-7
25	-f - g - x	-11	-7	-12	-9	-5	-6
26	-f - x - g	-7	-11	-5	-6	-12	-9
27	-x - f - g	-5	-6	-7	-11	-9	-12
28	f + g - x	-3	1	-2	1	3	4
29	f + x - g	1	5	-1	4	10	7
30	x + f - g	-1	4	1	5	7	10
31	f - g + x	5	1	10	7	-1	4
32	f - x + g	1	-3	3	4	-2	1
33	x - f + g	3	-2	5	9	1	4
34	f * g + x	10	6	11	8	4	5
35	f * x + g	6	10	4	5	29	17
36	x * f + g	4	5	6	10	17	29
37	f + g * x	22	10	11	8	4	5
38	f + x * g	10	22	4	5	11	8
39	x + f * g	4	5	6	10	8	11
40	f * g / x	3/7	1	4/7	1	3	4
41	f * x / g	1	7/3	1/3	4	28	16
42	x * f / g	1/3	4	1	7/3	16	28
43	f / g * x	7/3	1~	28	16	1/3	4
44	f / x * g	1~	3/7	3	4	4/7	1
45	x / f * g	3	1/4	9	21	1	7/4

表　1

No.	C	D	H	N	T
1	3	7	7	7	7
2	3	7	7	7	7
3	3	3	7	7	7
4	5	7	7	7	3
5	5	7	7	7	3
6	5	7	7	7	3
7	7	7	7	3	3
8	7	7	7	3	3
9	7	7	7	7	3
10	6	12	11	11	11
11	6	12	11	11	11
12	6	9	11	11	11
13	9	12	11	7	7
14	9	12	11	7	7
15	9	9	11	11	7
16	12	12	11	7	5
17	12	12	11	7	5
18	12	12	11	11	5
19	5	9	9	9	9
20	7	9	9	7	7
21	9	9	9	7	5
22	-3	-7	-7	-7	-7
23	-5	-7	-7	-5	-3
24	-7	-7	-7	-3	-3
25	-6	-12	-11	-11	-11
26	-9	-12	-11	-7	-7
27	-12	-12	-11	-5	-5
28	4	-2	-3	-3	-3
29	7	10	5	1	1
30	10	10	5	1	-1
31	4	10	5	5	5
32	1	-2	-3	1	1
33	4	4	9	5	3
34	5	11	10	10	10
35	17	29	10	6	6
36	29	29	10	6	4
37	5	8	22	22	22
38	8	8	22	22	10
39	11	11	10	10	4
40	4	0.571	0.429	0.429	0.429
41	16	28	2.333	1	1
42	28	28	2.333	1	0.333
43	4	28	2.333	2.333	2.333
44	1	0.571	0.429	1	1
45	1.75	1.75	21	9	3
46	17	32	28	24	12
47	20	32	28	24	12
48	2	4	12	12	12
49	53	95	28	18	18

表２　プログラム２の結果

VAL(10)＝F(J)＋G(J)＋X(J)

について，左から算術要素を1，2，3と番号づけすると次の6通りの値が得られる可能性がある．

(123)1＋3＋7＝11，(132)1＋3＋3＝7，(213)4＋1＋7＝12

(231)4＋1＋4＝9，(312)1＋3＋1＝5，(321)4＋1＋1＝6

このようにしてVAL(1)～(45)について評価の順序毎にあらかじめ計算した値を表1に示した．

このプログラムをC，D，H，N，Tの各処理系で計算した結果の一部(I＝1)を表2に示す．結果の多様性は予期以上であった．特定の処理系の結果を表1と照合することによって，各式毎の評価順序を決めてゆくことができる．これから解釈推論できる各処理系の算術式の評価方法の概略は次のようになる．

〔C〕 厳密に右から左に評価．

〔D〕 同じ強さの演算子に結合された関数が式中に複数個あれば右から左に評価する(多項演算子的)．たゞしカッコが優先する．

〔H〕 カッコの存在や演算子の優先順位には無関係に，式を左から右にみて，関数の1次子があれば(複数個あれば左から右の順に)評価して作業用番地に退避する．

〔N〕 カッコ，演算子の優先順位にしたがう．左から右の評価．単項演算子—が二項演算子—に優先して作用する．多項演算子的取扱いはない．

〔T〕 左から右．式中に同じ変数が2つ以上あれば，関数に先んじて評価される．

2．2 論理式の評価順序の検査

2．2．1 論理式と算術式

検査の立場からみて論理式と算術式の大きなちがいは次の2つ考えられる．

1）算術式は語の長さで表現しうるだけのいわば無数の値をとりうるのに反して，論理式は真偽の2値しかとりえない．したがって，算術式のように評価順序のちがいによってうまく値をちらすというだけでは評価順序の推測はむずかしい．

2）論理式の評価には必ずしもすべての項を評価する必要がない．JISも式の評価について「式の値を得るために必要に応じて式の一部が評価されるものとする．」と述べている(JIS p－10)．

各マニュアルの論理式の評価順序に関する記述は，要約すれば，ⅰ)算術式，ⅱ)関係式，ⅲ).NOT.，ⅳ).AND.，ⅴ).OR.の順に評価を行うと述べている．同じ順位の演算子については左から右の順に行うことを明記しているものといないものがある．この記述は2．1節1)の視点からは不十分で，とくに論理式中にあらわれる関数の評価順序ばかりでなく評価されない関数があれば，陽にあらわれた副作用はないとしても，その関数の実行によって変化を受けるはずの共通変数の値が変化しないことになり，プログラムの進行に影響をお

よぼし，プログラムの互換性に影響をもつ*　ここでは論理式の評価順序をしらべるために2つのプログラムを示す．

2．2．2　共通変数による方法

図2に共通変数の変化のちがいにより評価順序を知るプログラムを示す．このプログラムで添字は変数IND(1)～IND(20)は対応する論理式の値が真のとき0，偽のとき1の値をとるものとする．IND(1)～(10)に対応する論理式は論理関数LA(I)，LB(J)，LC(K)の.NOT.，.AND.，.OR.による結合でありこれらの関数には順に1，2，3の番号が割付けられ，引用される毎に共通変数の値に

IPERM＝IPERM＋N

という変化を与えることによって関数の評価順序が知れる．論理式の評価が終った時点で関数の評価順序がLC，LA，LB(3，1，2)となっていれば，IPERM＝312となっている．また部分評価が行なわれて，例えば第1式でLBとLCだけがこの順で評価された場合にはIPERM＝23となって，部分評価のあったことが知れる．この値は関数KPERMT(I)によってJND(I)に得られる．IND(11)～(20)に対応する論理式は，算術関数NA(I)，NB(J)，NC(K)を別々に含む関係式を1～10と同じように結合したもので，結合方法は1～10と11～20がこの順で対応している．

このプログラムの実行結果を表3に示す．当然のことながら1～10と11～20の結果は同じになるべきものであるが，一部にコンパイラのあやまりを検出してしまった．これから次のようなことがいえる

〔C，D〕右から左に全部評価．

〔N，T〕左から右に全部評価．

〔H〕左から右を原則とするが，部分論理式の値が確定すれば，その部分式に含まれる残りの項の評価は行なわず，他の部分式の評価にうつる．

2．2．3　副作用を利用した検査

算術式の場合と同じように副作用を利用したプログラムを図3に示す．2．2．1節に述べたように副作用を受けた結果も真偽の2値しかとりえないため結果の解釈は算術式に比べてむずかしい．このプログラムで論理関数LP(I)はI＝1のとき真，I＝2のとき偽を関数値とする．さらに副作用としては，共通変数JをI＝1のときJ＝2，I＝2のときJ＝1にそれぞれ反転させる．KND(I)が図2のIND(I)と同じ役割である．また

LP(J).AND.LP(J).AND.BL(J)

LP(J).OR.LP(J).OR.BL(J)

＊IBMシステム／360　FORTRANマニュアルでは論理式の中に評価されない関数がありうることを注意している(p. 27)

```
      SUBROUTINE BYLEXP
      LOGICAL BL,LP
      DIMENSION BL(2),KND(50)
      COMMON/BYLE/J
      BL(1) = .TRUE.
      BL(2) = .FALSE.
      WRITE(6,200)
      DO 1 I = 1, 2
         DO 2 K = 1, 50
    2    KND(K) = 1
         J = I
      IF( LP(J) .AND. BL(J) .AND. BL(J) ) KND(1)=0
         J = I
      IF((LP(J) .AND. BL(J)).AND. BL(J) ) KND(2)=0
         J = I
      IF( LP(J) .AND.(BL(J) .AND. BL(J))) KND(3)=0
         J = I
      IF( BL(J) .AND. LP(J) .AND. BL(J) ) KND(4)=0
         J = I
      IF((BL(J) .AND. LP(J)).AND. BL(J) ) KND(5)=0
         J = I
      IF( BL(J) .AND.(LP(J) .AND. BL(J))) KND(6)=0
         J = I
      IF( BL(J) .AND. BL(J) .AND. LP(J) ) KND(7)=0
         J = I
      IF((BL(J) .AND. BL(J)).AND. LP(J) ) KND(8)=0
         J = I
      IF( BL(J) .AND.(BL(J) .AND. LP(J))) KND(9)=0
         J = I
      IF(LP(J).AND.BL(J).OR. BL(J)) KND(10)=0
         J = I
      IF(BL(J).AND.LP(J).OR. BL(J)) KND(11)=0
         J = I
      IF(BL(J).AND.BL(J).OR. LP(J)) KND(12)=0
         J = I
      IF(LP(J).AND.LP(J).OR. BL(J)) KND(13)=0
         J = I
      IF(LP(J).AND.BL(J).OR. LP(J)) KND(14)=0
         J = I
      IF(BL(J).AND.LP(J).OR. LP(J)) KND(15)=0
         J = I
      IF(LP(J).OR. BL(J).AND.BL(J)) KND(16)=0
         J = I
      IF(BL(J).OR. LP(J).AND.BL(J)) KND(17)=0
         J = I
      IF(BL(J).OR. BL(J).AND.LP(J)) KND(18)=0
         J = I
      IF(LP(J).OR. LP(J).AND.BL(J)) KND(19)=0
         J = I
      IF(LP(J).OR. BL(J).AND.LP(J)) KND(20)=0
         J = I
      IF(BL(J).OR. LP(J).AND.LP(J)) KND(21)=0
         J = I
      IF(LP(J).OR. BL(J).OR. BL(J)) KND(22)=0
         J = I
      IF(BL(J).OR. LP(J).OR. BL(J)) KND(23)=0
         J = I
      IF(BL(J).OR. BL(J).OR. LP(J)) KND(24)=0
         J = I
      IF(.NOT.(LP(J).AND.BL(J)))    KND(25)=0
         J = I
      IF(.NOT.(BL(J).AND.LP(J)))    KND(26)=0
         J = I
      IF(.NOT.(LP(J).OR. BL(J)))    KND(27)=0
         J = I
      IF(.NOT.(BL(J).OR. LP(J)))    KND(28)=0
         J = I
      IF(.NOT.LP(J).OR. .NOT.BL(J)) KND(29)=0
         J = I
      IF(.NOT.BL(J).OR. .NOT.LP(J)) KND(30)=0
         J = I
      IF(.NOT.LP(J).AND..NOT.BL(J)) KND(31)=0
         J = I
      IF(.NOT.BL(J).AND..NOT.LP(J)) KND(32)=0
      WRITE(6,201) (K,K=1,32),(KND(K),K=1,32)
    1 CONTINUE
      RETURN
  200 FORMAT(1H /10X,
     145H*****SIDE EFFECT OF LOGICAL EXPRESSIONS*****   )
  201 FORMAT(1H0/(10X,3213))
      END
      LOGICAL FUNCTION LP( I )
      COMMON/BYLE/J
      LP = I.EQ.1
      J = 3 - I
      RETURN
      END
```

図2　プログラム2

処理系	論理関数の値	式番号（評価順序と論理式の値）									
	LA LB LC	1 11	2 12	3 13	4 14	5 15	6 16	7 17	8 18	9 19	10 20
C D	0 0 0	321 0	321 0	321 0	321 0	321 0	321 0	21 1	21 1	21 1	21 1
	0 0 1	321 1	321 0	321 0	321 0	321 1	321 0	21 1	21 1	21 1	21 1
	0 1 0	321 1	321 0	321 0	321 0	321 0	321 0	21 1	21 0	21 1	21 0
	0 1 1	321 1	321 1	321 1	321 0	321 1	321 0	21 1	21 0	21 1	21 0
	1 0 0	321 1	321 0	321 1	321 0	321 0	321 0	21 0	21 1	21 0	21 1
	1 0 1	321 1	321 1	321 1	321 1	321 1	321 0	21 0	21 1	21 0	21 1
	1 1 0	321 1	321 0	321 1	321 1	321 1	321 0	21 1	21 1	21 1	21 1
	1 1 1	321 1	321 1	321 1	321 1	321 1	321 1	21 1	21 1	21 1	21 1
	LA LB LC	1 11	2 12	3 13	4 14	5 15	6 16	7* 17*	8* 18*	9* 19*	10* 20*
N T	0 0 0	123 0	123 0	123 0	123 0	123 0	123 0	12 1	12 1	12 1	12 1
	0 0 1	123 1	123 0	123 0	123 0	123 1	123 0	12 1	12 1	12 1	12 1
	0 1 0	123 1	123 0	123 0	123 0	123 0	123 0	12 1	12 0	12 1	12 0
	0 1 1	123 1	123 1	123 1	123 0	123 1	123 0	12 1	12 0	12 1	12 0
	1 0 0	123 1	123 0	123 1	123 0	123 0	123 0	12 0	12 1	12 0	12 1
	1 0 1	123 1	123 1	123 1	123 1	123 1	123 0	12 0	12 1	12 0	12 1
	1 1 0	123 1	123 0	123 1	123 1	123 1	123 0	12 1	12 1	12 1	12 1
	1 1 1	123 1	123 1	123 1	123 1	123 1	123 1	12 1	12 1	12 1	12 1
	LA LB LC	1	2	3	4	5	6	7	8	9	10
H	0 0 0	123 0	12 0	12 0	1 0	13 0	1 0	1 1	12 1	1 1	12 1
	0 0 1	123 1	12 0	12 0	1 0	13 1	1 0	1 1	12 1	1 1	12 1
	0 1 0	12 1	123 0	123 0	1 0	13 0	1 0	1 1	12 0	1 1	12 0
	0 1 1	12 1	123 1	123 1	1 0	13 1	1 0	1 1	12 0	1 1	12 0
	1 0 0	1 1	13 0	1 1	123 0	123 0	12 0	12 0	1 1	12 1**	1 1
	1 0 1	1 1	13 1	1 1	123 1	123 1	12 0	12 0	1 1	12 1**	1 1
	1 1 0	1 1	13 0	1 1	12 1	12 1	123 0	12 1	1 1	12 0**	1 1
	1 1 1	1 1	13 1	1 1	12 1	12 1	123 1	12 1	1 1	12 0**	1 1
	LA LB LC	11	12	13	14	15	16	17	18	19	20
	0 0 0	123 0	12 0	12 0	1 0	13 0	1 0	1 1	12 1	1 1	12 1
	0 0 1	123 1	12 0	12 0	1 0	13 1	1 0	1 1	12 1	1 1	12 1
	0 1 0	12 1	123 0	123 0	1 0	13 0	1 0	1 1	12 0	1 1	12 0
	0 1 1	12 1	123 1	123 1	1 0	13 1	1 0	1 1	12 0	1 1	12 0
	1 0 0	1 1	13 0	1 1	123 0	123 0	12 0	12 0	1 1	12 0	1 1
	1 0 1	1 1	13 1	1 1	123 1	123 1	12 0	12 0	1 1	12 0	1 1
	1 1 0	1 1	13 0	1 1	12 1	12 1	123 0	12 1	1 1	12 1	1 1
	1 1 1	1 1	13 1	1 1	12 1	12 1	123 1	12 1	1 1	12 1	1 1

注　* T では .NOT. を含む論理式のコンパイルが正常に動作しないので，この結果は N のみのもの．

** 論理関数と .NOT. の結合であやまったコンパイルをしている．同じ論理値をもつ関係式(19)では正常．

表3　プログラム2の結果

```
      SUBROUTINE LOGEXP
      LOGICAL LA,LB,LC
      COMMON/PERMUT/IPERM
      DIMENSION IND(20),JND(20)
      IPERM =0
      WRITE(6,100) (L,L=1,10)
      DO 2 I= 1,2
      DO 2 J= 1,2
      DO 2 K = 1,2
          DO 1 L = 1,10
          IND(L) = 1
    1     JND(L) = 0
      IF( LA(I) .AND. LB(J) .AND. LC(K) )    IND( 1)=0
      JND( 1) = KPERMT( 1)
      IF( LA(I) .AND. LB(J) .OR.  LC(K) )    IND( 2)=0
      JND( 2) = KPERMT( 2)
      IF( LA(I) .AND.(LB(J) .OR.  LC(K)))    IND( 3)=0
      JND( 3) = KPERMT( 3)
      IF( LA(I) .OR.  LB(J) .AND. LC(K) )    IND( 4)=0
      JND( 4) = KPERMT( 4)
      IF((LA(I) .OR.  LB(J)).AND. LC(K) )    IND( 5)=0
      JND( 5) = KPERMT( 5)
      IF( LA(I) .OR.  LB(J) .OR.  LC(K) )    IND( 6)=0
      JND( 6) = KPERMT( 6)
      IF( .NOT.LA(I).AND.LB(J) )             IND( 7)=0
      JND( 7) = KPERMT( 7)
      IF( LA(I) .AND. .NOT. LB(J))           IND( 8)=0
      JND( 8) = KPERMT( 8)
      IF( .NOT.(LA(I) .OR. .NOT.LB(J)))      IND( 9)=0
      JND( 9) = KPERMT( 9)
      IF( .NOT.(.NOT.LA(I).OR. LB(J) ) )     IND(10)=0
      JND(10) = KPERMT(10)
      II = I - 1
      JJ = J - 1
      KK = K - 1
      WRITE(6,101) II,JJ,KK,(JND(L),IND(L),L=1,10)
    2 CONTINUE
      WRITE(6,100) (L, L=11,20)
      DO 4 I = 1,2
      DO 4 J = 1,2
      DO 4 K = 1,2
          DO 3 L = 11,20
          IND(L) = 1
    3     JND(L) = 0
      IF( 2*NA(I).LT.3 .AND. 2*NB(J).LT.3 .AND. 2*NC(K).LT.3 )IND(11)=0
        IND(12) ～ IND(19)  は 中略.
      IF(.NOT.(.NOT.(2*NA(I).LT.3) .OR. 2*NB(J).LT.3)) IND(20)=0
      JND(20) = KPERMT(20)
      II = I - 1
      JJ = J - 1
      KK = K - 1
      WRITE(6,101)II,JJ,KK,(JND(L),IND(L),L=11,20)
    4 CONTINUE
      RETURN
  100 FORMAT(1H0/10X,10H  I  J  K ,1017/)
  101 FORMAT(10X,3I3,2X,10(I5,I2))
      END
```

```
      FUNCTION KPERMT( I )
      COMMON/PERMUT/ IPERM
      KPERMT = IPERM
      IPERM = 0
      RETURN
      END
      LOGICAL FUNCTION LA( I )
      LOGICAL LZ
      LA = LZ( I, 1)
      RETURN
      END
      LOGICAL FUNCTION LB( J )
      LOGICAL  LZ
      LB = LZ( J, 2)
      RETURN
      END
      LOGICAL FUNCTION LC( K )
      LOGICAL  LZ
      LC = LZ( K, 3)
      RETURN
      END
      LOGICAL FUNCTION LZ(I, N)
      COMMON/PERMUT/IPERM
      IPERM = IPERM*10+N
      LZ = I.E@.1
      RETURN
      END
      FUNCTION NA( I )
      NA = NZ( I, 1)
      RETURN
      END
      FUNCTION NB( J )
      NB = NZ( J, 2)
      RETURN
      END
      FUNCTION NC( K )
      NC = NZ( K, 3)
      RETURN
      END
      FUNCTION NZ(I, N)
      COMMON/PERMUT/IPERM
      IPERM = IPERM * 10 + N
      NZ = I
      RETURN
      END
```

図3　プログラム3

などは副作用を受けてもすべての評価順序の組み合せについて論理値が同一になるので省略してある．（25）～（28）式にDeMorganの法則の適用したのが（29）～（32）式で，この順に対応している．算術式の場合の表1に対応する評価順序による論理値の表をあらかじめ作って，表4に示しておく，この表では副作用によって変化のあるものだけをあげた．このプログラムの実行の結果を表5に示す．

結果の解析の作業は2．2．2節の結果から，全ての論理要素が評価されるとみなされるC，D，N，Tでは，表4の該当する値の場所と照合することによって，その中から評価順序のたゞ一つにきまるものから手掛りにする．

〔N〕 評価順序が一意的にきまるのは(25)～(32)と(14)，(15)式である（それぞれ123，213）．これから(4)，(23)式以外は順序を一意に推定できる．これらが対応する算術式の場合と全く同じであるところから，逆に(4)，(23)の順序も算術式VAL(4)の場合と同じ順序と仮定する．

〔D〕 (25)～(32)，(19)，(20)が一意的．Nの場合の左右の関係を完全に入れかえたものであることがわかる．(1)，(4)，(22)，(23)を除いてNと同様に順序の推定ができる．(1)，…，(23)は算術式の結果から多項演算子的取扱いを仮定する．

〔C〕 (25)～(32)，(17)，(20)式が一意的．算術式と同様，完全に右から左の評価．

〔T〕 .NOT. が不備のため判定不能．

〔H〕 前節の結果により十分な情報を得ている．(25)～(32)，(14)式が一意的．算術式の場合とは関数の取扱いがちがうことに注意．

3．浮動小数点演算の丸め誤差の振るまい

3．1 浮動小数点演算の単位丸め誤差の検査

3．1．1 単位丸め誤差の定義

浮動小数点演算の丸め誤差の処理方式について精確に言及している言語マニュアルはほとんどないといってよい．

浮動小数点演算の単位丸め誤差は，高級言語を介した処理系毎に異るか，この誤差をForsytheら[3]は，1.0に対する相対誤差として論理式

$$1.0 + \mathrm{EPS} = 1.0 \qquad (1)$$

を満たすようなEPSの最大値と定義している．今二進法の計算機だけを考えEPSは，次のような集合に属する数とする

$$\{\pm 2^{-i}\} \quad (i > 0) \qquad (2)$$

浮動小数点数の仮数部の長さをt桁（符号部を除く）とすると(1)式を満足するこの集合の最大値は

0捨1入演算で $\mathrm{EPS} = 2^{-t-1}$ (3)

切捨て演算で $\mathrm{EPS} = 2^{-t}$ (4)

式番号	論理式	J−1	評価順序					
			123	132	213	231	312	321
1	L ∧ B ∧ B	0	1	1	1	0	1	0
2	(L ∧ B) ∧ B	0	1	1	1	0	1	0
3	L ∧ (B ∧ B)	0	1	1	1	0	1	0
4	B ∧ L ∧ B	0	1	0	1	1	0	1
5	(B ∧ L) ∧ B	0	1	0	1	1	0	1
6	B ∧ (L ∧ B)	0	1	0	1	1	0	1
7	B ∧ B ∧ L	0	0	1	0	1	1	1
8	(B ∧ B) ∧ L	0	0	1	0	1	1	1
9	B ∧ (B ∧ L)	0	0	1	0	1	1	1
10	L ∧ B ∨ B	0	1	1	0	0	0	0
		1	0	0	0	1	1	1
11	B ∧ L ∨ B	0	0	0	1	1	0	0
		1	0	1	0	0	1	1
12	B ∧ B ∨ L	1	1	1	1	1	0	0
13	L ∧ L ∨ B	0	0	1	0	1	0	0
		1	1	0	1	0	1	1
14	L ∧ B ∨ L	0	1	0	0	0	0	0
		1	0	0	0	1	1	0
15	B ∧ L ∨ L	0	0	0	1	0	0	0
		1	0	1	0	0	0	1
16	L ∨ B ∧ B	1	0	0	1	1	1	1
17	B ∨ L ∧ B	0	0	0	1	1	0	0
		1	1	1	0	0	1	0
18	B ∨ B ∧ L	0	0	0	0	0	1	1
		1	1	1	1	0	0	0
19	L ∨ L ∧ B	0	0	0	0	1	0	0
		1	1	0	0	0	1	0
20	L ∨ B ∧ L	0	0	0	0	0	0	1
		1	0	1	1	0	0	0
21	B ∨ L ∧ L	0	0	0	1	0	1	0
		1	1	1	0	1	0	1
22	L ∨ B ∨ B	1	0	0	0	1	0	1
23	B ∨ L ∨ B	1	0	1	0	0	1	0
24	B ∨ B ∨ L	1	1	0	1	0	0	0

式番号	論理式	J−1	評価順序	
			12	21
25	¬(L∧B)	0	0	1
26	¬(B∧L)	0	1	0
27	¬(L∨B)	1	1	0
28	¬(B∨L)	1	0	1
29	¬L ∨ ¬B	0	0	1
30	¬B ∨ ¬L	0	1	0
31	¬L ∧ ¬B	1	1	0
32	¬B ∧ ¬L	1	0	1

表 4

注1. ¬∧∨はそれぞれ.NOT. .AND. .OR. を示す.
2. L, Bは LP(J), BL(J) を示す.
3. J−1は副作用のない場合のLP, BLの値.
4. 評価順序は論理式中の論理要素を左から1, 2, 3と番号をつける.
5. 0が真, 1が偽をあらわす.

式番号	J−1	処理系				
		C	D	H	N	T
1	0	0	1	1	1	1
2	0	0	1	1	1	1
3	0	0	0	1	1	1
4	0	1	1	1	1	0
5	0	1	1	1	1	0
6	0	1	1	1	1	0
7	0	1	1	0	0	0
8	0	1	1	0	0	0
9	0	1	1	0	1	0
10	0	0	1	1	1	1
	1	1	0	0	0	0
11	0	0	1	0	1	0
	1	1	0	1	0	1
12	1	0	0	1	1	1
13	0	0	0	0	0	0
	1	1	1	0	1	1
14	0	0	0	1	1	1
	1	0	1	0	0	0
15	0	0	0	0	1	0
	1	1	1	1	0	0
16	1	1	1	0	0	0
17	0	0	1	0	1	0
	1	0	0	1	0	1
18	0	1	1	0	1	0
	1	0	0	1	0	1
19	0	0	1	0	0	0
	1	0	0	1	1	1
20	0	1	1	0	0	0
	1	0	0	0	1	0
21	0	0	0	0	0	0
	1	1	1	1	1	1
22	1	1	0	0	0	0
23	1	0	0	0	0	1
24	1	0	0	1	1	1
25	0	1	0	0	0	*
26	0	0	0	1	0	*
27	1	0	1	1	1	*
28	1	1	1	0	1	*
29	0	1	1	0	0	*
30	0	0	0	1	1	*
31	1	0	0	1	1	*
32	1	1	1	0	0	*

＊は .NOT. コンパイルにエラーがあり結果が得られなかった.

表5 プログラム3の結果

となることが予期される（これらは，それぞれの方式の単位丸め誤差の１／２である）．ところでFORTRANでこれらの値を知るプログラムを書くには，いろいろ面倒な問題に遭遇する．自然に考えれば，

IF(1.0 + 2^{-i} .EQ. 1.0) GO TO 2　　（５）

IF(1.0 + 2^{-i} − 1.0) 1,2,3　　（６）

について，iを大きくしていって最初に文番号２に脱出したときのiの値が（３）または（４）のEPSを与える（2^{-i}は便宜的記述である）．しかし，個々の処理系のハードウエアおよびコンパイラの処理方式のちがいによって，このプログラムは必ずしも期待する結果を我々に与えてくれない．このように互換性を害う要因はほゞ次のように考えられる．

１）浮動小数点演算をとり扱うAccの長さが必ずしもt桁とは限らない．

２）実数型と実数型の演算結果が実数型になるもの（JISの規定），と倍精度実数型になるものがある．

３）絶対値表示と補数表示では，負数の場合と正数の場合で丸められた値の代表する区間が異る．

４）４捨５入方式（２進法では０捨１入）と切捨て方式があり，４捨５入方式の採用はimplementerに依存する場合が多い．

５）組込み関数SNGLおよびDBLEの動作点に関する言語マニュアルのあいまいな記述．1)との関連．さらに単精度化の際の丸め方式．

６）Accが複数個ある場合と１個の場合の中間結果の処理．

７）関係式のコンパイル方法の多様性．

８）最適化の程度．例えば（５）の文の関係式の両辺からコンパイル時に定数1.0を消去してしまっては，実行時に（５）の文は絶対にtrueになりえない．

３．１．２　論理IFと算術IFの実行

（５）式と（６）式の互換性をしらべるために（５），（６）のいくつかの書きかえを作ってその動作をしらべてみた．さらに８）の効果がおよばないように

IF(A+EPS.EQ. B)　　（７）

も加えた（A＝B＝1.0または−1.0）．すなわち次の３つの場合を（２）の集合に属するEPSについて実行させた．

Case 1　1.0 + EPS = 1.0

Case 2　A + EPS = B （A＝B＝1.0）

Case 3　A + EPS = B （A＝B＝−1.0）

プログラムの一部を図４に示す．これには，８）の効果をしらべるために

IF(1.0 + EPS .EQ. 1.0 + EPS)

IF(A + EPS .EQ. B + EPS)

```
      DO 1 I=1,20
    1 IND(I)=1
      IF(1.0+EPS.EQ.1.0)              IND(1) = 0
      IF(EPS+1.0.EQ.1.0)              IND(2) = 0
      IF(1.0.EQ.1.0+EPS)              IND(3) = 0
      IF(1.0.EQ.EPS+1.0)              IND(4) = 0
      IF(SNGL(1.0+DEPS).EQ.1.0)       IND(5) = 0
      IF(1.0.EQ.SNGL(1.0+DEPS))       IND(6) = 0
      IF(1.0+EPS.EQ.1.0+EPS)          IND(7) = 0
      INDEX = 8
1008  IF(1.0+EPS-1.0)                 12,13,14
1009  IF(EPS+1.0-1.0)                 12,13,14
1010  IF(1.0-(1.0+EPS))               12,13,14
1011  IF(1.0-(EPS+1.0))               12,13,14
1012  IF(SNGL(1.0+DEPS)-1.0)          12,13,14
1013  IF(1.0-SNGL(1.0+DEPS))          12,13,14
1014  IF(1.0+EPS-(1.0+EPS))           12,13,14
1015  IF(DBLE(SNGL(1.0+DEPS)-1.0))    12,13,14
1016  IF(DBLE(1.0-SNGL(1.0+DEPS)))    12,13,14
   12 IND(INDEX) = -1
      GO TO 15
   13 IND(INDEX) = 0
      GO TO 15
   14 IND(INDEX) = 1
   15 INDEX = INDEX+1
      KEY=INDEX-8
      GO TO(1009,1010,1011,1012,1013,1014,1015,1016,30),KEY
```

図4　プログラム4（case1）

項目＼処理系	C	D	H	N	T
1語の長さ（ビット）*	36	36	32	48	48
単精度仮数部 *	26	26	23	35	37
1） Accの長さ	R′ **	R′	R′ （＝D）	R	R
2） *	R・R＝R	R・R＝R	R・R ＝ D	R・R ＝ R	R・R ＝ R
3）	補数表示	同　左	同　左	同　左	同　左
4）丸めの方式	0捨1入	同　左	単精度については不明	切　捨　て	同　左
5）SNGLの動作	Dを切捨ててR′にする.	同　左	Dを切捨ててRにする.	同　左	Dを0捨1入してRにする
6）Accの個数	1個(中間結果の退避法から)	同　左	複　数　個 *	マニュアルに記述なし	同　左
8）定数の消去	な　し	なんともいえない	な　し	な　し	な　し
共通式のくくり出し	な　し	あ　り	R・R＝Dのため不明	切捨て演算のため不明	同　左

＊　事前に言語マニュアルから知れる情報.

＊＊　R′はAccがt桁をこえることを意味する.

表6　単精度の結果から得られる情報

がつけ加えてある．紙数の都合で結果は省略する．この結果から推測できる丸め誤差の処理に関する情報の一部を表6にまとめた．この表で特に組込み関数SNGLの多様性に注目されたい．なお同様な計算を倍精度についても行った．これは主として2）のためである．

3．2 丸め誤差を利用した評価順序の検査

3．2．1 準 備

2．1節では副作用を利用して算術式の評価順序の検査を行った．ここでNについてはかなりの情報が得られたが，C，Dについては右から左に関数の評価を行なっていながら，マニュアルには，同一演算子に対しては左から右の順へ演算を行うという記述がある．そこで2．1節2）の視点を逆用して，浮動小数点演算で，中間結果が一たん作業用番地に退避されるときに一般に丸め誤差が発生することから，データを適当に与えることによって，ある項が他の項に先だって評価されたことを知ることができることを利用する．

次のような3項からなる算術式を考える．

$$\pm A\times B\pm A\times C\pm A\times D = V \qquad (1)$$

ここで複号は全ての組合せをとることにすると8通りの式が得られる．複号を順に（＋＋＋），（＋＋－），（＋－＋），（＋－－），（－＋＋），（－＋－），（－－＋），（－－－）を変えた式を$V_1, V_2, \cdots, V_8$とする．ここでも，2進法の計算機のみを考えることにすると前節と同様に仮数部の長さをt桁（符号を除く）として，$\Delta^0 = 2^{-t+1}$，$\Delta^{-1} = 2^{-t}$，$\Delta^{-2} = 2^{-t-1}$ 一般に

$$\Delta^{-i} = 2^{-t+1-i} = \Delta^{-i-1} + \Delta^{-i-1} \qquad (2)$$

とおくと

$$2 - \Delta^0 \geq |V| \geq 1 + \Delta^0 \qquad (3)$$

の浮動小数点数について単位丸め誤差の大きさは共通に，0捨1入方式ではΔ^{-1}，切捨て方式ではΔ^0となる．いま

$$\left.\begin{aligned} &A = 1+\alpha,\ B = 1+\beta,\ C = r,\ D = \delta \\ &\alpha = 2^{-a},\ \beta = 2^{-b},\ r = 2^{-c},\ \delta = 2^{-d} \\ &a, b, c, d \text{は正整数} \end{aligned}\right\} \qquad (4)$$

のようにA，B，C，Dを選びいずれの丸め方式をとる場合にも通用するように（3）式を満たすようにα，β，r，δを選ぶ．たゞし

$$\alpha\beta, \alpha r, \alpha\delta = \Delta^0,\ \Delta^{-1} \text{または} \Delta^{-2} \qquad (5)$$

となるようにする．結局

$$1 - 4\cdot 2^{-t+1} > 2^{-a} + 2^{-b} \pm 2^{-c} \pm 2^{-d} > 3\cdot 2^{-t+1} - 2^{-t} \qquad (6)$$

をみたすようなa，b，c，dを求める．いま

$$a = [t/3] \qquad (7)$$

とおくと

$u_0 = t-1, u_1 = t, u_2 = t+1$ (8)

として，(5)式から

$b = u_j - a, c = u_k - a, d = u_l - a$ (9)

$(j, k, l = 0, 1, 2)$

を代入すれば，通常の計算機の浮動小数点数に対して(6)式が成立している．a，b，c，dをこのように選ぶと$A \times B = 1 + \alpha + \beta + \alpha\beta$，$A \times C = r + \alpha r$，$A \times B = \delta + \alpha\delta$ から，A×CとA×Dはt桁で完全に表現できるのでこれらが中間結果として退避されても丸め誤差を発生しない．一方A×Bはt，$t+1$，$t+2$桁となるので，後2者のとき次のような丸め誤差が発生する．

0捨1入で$\alpha\beta = \Delta^{-1}$のとき　$A \times B = 1 + \alpha + \beta \fallingdotseq \Delta^{0}$

$\alpha\beta = \Delta^{-2}$のとき　$A \times B = 1 + \alpha + \beta$

切捨てではともにA×B＝1＋α＋β

さてこれまでの考察では，B＝1＋βと選んだのでA×Bが他の2項に優越しているが，

$B = \beta$，$C = 1 + r$，$D = \delta$ (10)

$B = \beta$，$C = r$，$D = 1 + \delta$ (11)

としても同じ議題が展開できるので，(10)では，A×Cを(11)ではA×Dを優越項とすることができる．そしてこの優越項がもし中間結果として作業用番地に退避されれば，そのとき発生する丸め誤差によって算術式の評価順序に関する情報が得られることになる．

3．2．2　プログラムと結果

プログラムを図5に示した．VAL(1)～(8)がV_1～V_8に対応する(倍精度ではBAL(1)～(8))．サブルチンDUMMYは，コンパイラの最適化操作を逃れるために挿入した実行上は無意味なサブルチンである．したがって，VAL(9)～(28)は最適化(共通式のくくり出し)，カッコの有無，関数の有無の影響を検知する目的をもっている．共通変数BASISは，$BASIS(I) = 2^{-I}$，LL，MM，NNはそれぞれ1語長のビット数，単精度浮動小数点数仮数部のビット数(符号を除く)，倍精度浮動小数点数のビット数(符号を除く)が与えられている．サブルチンTHREEのパラメタ，IA，IB，IC，IDはそれぞれ，a，b，c，dに対応し，KEYは1，2，3の値をとり，この順に(4)，(10)，(11)式に対応している(これらをcase 1，case 2，case 3とする)．

このプログラムを各処理系で計算した結果の一部(V_1，V_2，V_7)を表7に示す．これは計算結果の最後の3ビットを8進数で示したものである．また誤差項の0，−1，−2はそれぞれΔ^{c}，Δ^{-1}，Δ^{-2}を略記した．さらに，表6からテストに用いた各処理系はすべて補数表示である．C，Dでは仮数部を処理するAccの長さが仮数部より長いこと，中間結果の退避に際して0捨1入が行われていることを考慮すると(表6より)，C，Dの評価順序は，

```
      SUBROUTINE THREE(IA,IB,IC,ID,KEY)
      COMMON BASIS(100),LINE
      COMMON/SORD/KK,LL,MM,NN,IBASE
      DOUBLE PRECISION E,F,G,H,Y,S,BAL
      DIMENSION VAL(60),BAL(30),R(3),S(3),II(3)
      EQUIVALENCE (B,R(1)),(C,R(2)),(D,R(3)),
     1 (F,S(1)),(G,S(2)),(H,S(3)),(VAL,BAL),
     2 (J,II(1)),(K,II(2)),(L,II(3))
      I = IA
      J = IB
      K = IC
      L = ID
      IF(KK-NN) 1,2,2
C*****SINGLE PRECISION*****
    1 CONTINUE
      A = BASIS(I) + 1.0
      B = BASIS(J)
      C = BASIS(K)
      D = BASIS(L)
      R(KEY) = R(KEY) + 1.0
      II(KEY) = -II(KEY)
      I = -I
      VAL( 1)=  A * B + A * C + A * D
          CALL DUMMY(B,D)
      VAL( 2)=  A * B + A * C - A * D
          CALL DUMMY(B,D)
      VAL( 3)=  A * B - A * C + A * D
          CALL DUMMY(B,D)
      VAL( 4)=  A * B - A * C - A * D
          CALL DUMMY(B,D)
      VAL( 5)=- A * B + A * C + A * D
          CALL DUMMY(B,D)
      VAL( 6)=- A * B + A * C - A * D
          CALL DUMMY(B,D)
      VAL( 7)=- A * B - A * C + A * D
          CALL DUMMY(B,D)
      VAL( 8)=- A * B - A * C - A * D
          CALL DUMMY(B,D)
      VAL( 9)=  A * B + A * C + A * D
      VAL(10)=  A * B + A * C - A * D
      VAL(11)=  A * B - A * C + A * D
      VAL(12)=  A * B - A * C - A * D
      VAL(13)= A   *X(J) + A   * C   + A   *D
      VAL(14)=(A   *X(J) + A   * C  )+ A   *D
      VAL(15)= A   *X(J) +(A   * C   + A   *D  )
      VAL(16)= A   * B   + A   *X(K) + A   *D
      VAL(17)=(A   * B   + A   *X(K))+ A   *D
      VAL(18)= A   * B   +(A   *X(K) + A   *D  )
      VAL(19)= A   * B   + A   * C   + A   *X(L)
      VAL(20)=(A   * B   + A   * C  )+ A   *X(L)
      VAL(21)= A   * B   +(A   * C   + A   *X(L))
      VAL(22)= X(I)*X(J) + A   * C   + A   *D
      VAL(23)= A   * B   + X(I)*X(K) + A   *D
      VAL(24)= A   * B   + A   * C   + X(I)*X(L)
      VAL(25)= X(I)*X(J) + X(I)*X(K) + A   *D
      VAL(26)= X(I)*X(J) + A   * C   + X(I)*X(L)
      VAL(27)= A   * B   + X(I)*X(K) + X(I)*X(L)
      VAL(28)= X(I)*X(J) + X(I)*X(K) + X(I)*X(L)
      CALL EJECT(30)
      WRITE(6,100) IA,IB,IC,ID,KEY
      WRITE(6,101) (I,VAL(I),I,VAL(I),I=1,28)
      RETURN
C*****DOUBLE PRECISION*****
    2 CONTINUE
      E = DBLE(BASIS(I)) + 1.0D0
      F = DBLE(BASIS(J))
      G = DBLE(BASIS(K))
      H = DBLE(BASIS(L))
      S(KEY) = S(KEY) + 1.0D0
      II(KEY) = -II(KEY)
      I = -I
      BAL( 1)=  E * F + E * G + E * H

              (  中      略  )

      BAL(28)= Y(I)*Y(J) + Y(I)*Y(K) + Y(I)*Y(L)
      CALL EJECT(30)
      WRITE(6,100) IA,IB,IC,ID,KEY
      WRITE(6,102) (I,BAL(I),I,VAL(2*I-1),VAL(2*I),I=1,28)
      RETURN
  100 FORMAT(1H /,7X,2HI=,I2,6H  ( J=,I2,3H,K=,I2,3H,L=,I2,
     124H)  THE DOMINANT TERM IS ,I5,1H. )
  101 FORMAT(I9,E31.21,I5,1X,O16)
  102 FORMAT(I9,E31.21,I5,1X,O16,1X,O16)
      END

      FUNCTION X(I)
      COMMON BASIS(100)
      IF(I) 1,1,2
    1 J = -I
      X = 1.0 + BASIS(J)
      RETURN
    2 X = BASIS(I)
      RETURN
      END

      DOUBLE PRECISION FUNCTION Y(I)
      COMMON BASIS(100)
      IF(I) 1,1,2
    1 J = -I
      Y = 1.0D0 + DBLE(BASIS(J))
      RETURN
    2 Y = DBLE(BASIS(I))
      RETURN
      END
```

図5　プログラム5

式	V_1=A*B+A*C+A*D				V_2=A*B+A*C−A*D				V_7=−A*B−A*C+A*D			
処理式	CD	H	N	T	CD	H	N	T	CD	H	N	T
誤差項	CASE	CASE	CASE	CASE	CASE	CASE	CASE	CASE	CASE	CASE	CASE	CASE
αβ αr αδ	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3	1 2 3
0 −1 −1	2 ③③	0 1 1	1 1 1	1 1 1	1 ②1	0 1 1	1 1 2	0 0 1	7 7 ①	6 7 6	7 7 6	6 7 6
−1 0 −1	② 2 ③	1 0 1	1 1 1	1 1 1	① 1 1	1 0 1	1 1 2	0 0 1	7 7 0	5 7 6	6 7 6	7 6 6
−1 −1 −0	②③ 2	1 1 1	1 1 2	1 1 2	⓪① 0	7 7 1	7 7 0	1 7 0	0 0 0	7 0 7	0 1 0	0 0 0
−1 −1 −1	2 2 2	0 0 1	0 0 1	0 0 1	1 1⓪	0 0 1	0 0 1	7 7 1	⑦⑦ 0	6 7 7	7 0 7	7 7 7
−1 −1 −2	①② 1	0 0 1	0 0 1	0 0 1	1 1 1	0 0 1	0 0 1	7 7 1	7 7 7	6 7 7	7 0 7	7 7 7
−1 −2 −1	① 1 ②	0 0 0	0 0 0	0 0 0	⓪ 0 0	0 0 1	0 0 1	7 7 0	⑦ 0 0	6 7 7	7 0 7	7 7 7
−2 −1 −1	1 ②②	0 0 0	0 0 0	0 0 0	0 ① 0	0 0 1	0 0 1	7 7 0	0 ⑦ 0	6 7 7	7 0 7	7 7 7

Hのみ倍精度計算の結果

表7　V_1 , V_2 V_7 の仮数部最後の3ビットの値

V_1 ：A×B＋A×C＋A×D　　V_7 ：−A×B−A×C＋A×D

③ ② ①

④

⑤

⑥

が知れる．表7のDの結果のうち◯で囲んだものが上記の推論のかぎとなったものである．

N , Tでは，Accの長さと仮数部の長さが等しく計算の結果は退避されなくともつねに切捨てられる（表6より）．このため評価順序に関する情報は十分保存されないので，この方法は必ずしも有力ではないが，Case 1とCase 2 , 3の比較から，同じ順位の演算子について左から右に演算が行われることは知れる．

最後にHでは，R∘R＝Dとなるので表6の結果は倍精度演算によったものである．この場合も倍精度で切捨てが行われるのでNと同じコメントが与えられる．H , N , T の仮数部の最後の桁における1ビット程度の差は，Accの丸め誤差処理回路の余裕の有無が原因と思われる．

次に共通式のくくり出しを行っているのはDのみである（VAL(9) ～ (12)の結果から）のでDについてVAL(13) ～ (28)の結果のみをあげる（表8）．

丸め誤差の問題は，単にソフトウェアだけでなく，ハードウェア設計上の問題もからむため，このプログラムの結果は，式の評価の問題をはなれて興味深い．

3．3　複素数演算の組込み関数について

FORTRANの中で複素数演算は，整数や実数演算に比較してよりソフト的で，演算中に生ずる全ての過程を利用者が記述しなくともよいので，必要な場合には，かなりの利用者があるようである．ところで，丸め誤差のために，×を複素変数とすると×＊CONJG(×)はCABS(×)の平方に等しくなることはほとんどありえない．そのほか，複素数の簡単な四則も，利用者の目の届かないところで，中間結果の退避や丸めが行われ利用者の予期しないような丸め誤差やケタ落ちが生ずることがある．そこでこれらの振るまいを知るために3．1節のようなプログラムを作成してしらべてみた．ここでは紙数の関係で省略する．

式番号 / $\alpha\beta\ \alpha r\ \alpha\delta$	13 14 22 25 26 28	15	16 17 18	19 20 24	21	23 27
0 −1 −1	2 3 3		2	3 2	2	2
−1 0 −1	2 2 3	3	3	3 3 2	2	3
−1 −1 0	2 3 2	3	3 2	3	3	3 2
−1 −1 −1	2 2 2	3 3		3 3	3	
−1 −1 −2	1 2 1	2	2 1	2	2	2 1
−1 −2 −1	1 1 2	2	2	2 2 1	2 1	2
−2 −1 −1	1 2 2		1	2 1	1	1

空白は(13)の対応する欄に同じ

表8　共通式のくくり出しの影響(D)

4。あとがき

プログラムとその結果は他の処理系でも検査できるようにできるかぎり完全な形で示すことに努めたが，実行の結果の解釈は紙数の都合で詳細に記すことができなかった．ところでここにあげたプログラムはコンパイラを含んだ処理系を一種のブラックボックスとしてそのパラメタを完全に決定する目的では十分ではない．しかしあるコンパイル方法が与えられて，特定のコンパイラがその方法をとっているかどうかを決定する問題として，コンパイル方法ごとに検査プログラムを作成することができたとしても，利用者の立場からテストとしては，簡便さが要求されるので，このような一種の決定問題の解として処理系のパラメタを探すのでは，実用性が薄くなる．ここに述べたようにたまたま筆者が利用できた5つの処理系だけでもこのように多様な方法をとっており，現在のコンパイラ技術では人間のテストをしているようなことになってしまう．コンパイラコンパイラなどが実用化された時点での問題として興味あるものと思う．

なおこの計算は，九大，東大，阪大の各大型計算機センターと京大数理研所属の各計算機〔FACOM230−60 FORTRAN−CおよびD，HITAC5020 E HARP，NEAC2200−500 FORTRAN−L，RIMS−I(TOSBAC3400−30) FORTRAN 7000〕を使用した．関係各位に謝意を表する．本文中で言及したマニュアルは，上記計算機のFORTRAN

マニュアルである.

文　献

1）電子計算機プログラム用言語FORTRAN(水準7000) JISC6201－1967日本規格協会PP.62(1967)

2）牛島和夫：FORTRAN語の拡張と互換性について，情報処理Vol.11,No.8.P.484(1970)

3）G.E.Forsythe and C.E.Moler:Computer Solution of Linear Algebraic Systems, Prentice Hall PP.148(1967)