≪解　説≫

教育用主記憶常駐型 FORTRAN コンパイラの現状

土　居　範　久*

1. まえがき

大学における計算機教育は，専門の学科も既にいくつか設置され着々と進展しつつある．なかでもプログラミング（あるいはアルゴリズム）教育は次第に広く行なわれようとしている．対象とする学生も理工学部にとどまらず次第に全学的に広がっている．たとえば，東京大学や慶応義塾大学ではカフェテリア方式を用いてほぼ全学的に行なわれているし，筑波大学では全学的な必修科目のひとつとして計算機教育がとり上げられている．

教育用ジョブは，プログラム作成者が未熟である，ボリュームが小さい，実行時間が短いといった特徴がある．そこで，大学の計算センターで教育用にかなり使用しているところでは，大部分のジョブは極めて短いプログラムと思われる．しかし，融通性をもたせ，かつじゅうぶん最適化する機能をもたせながら，場合によっては最適化は行なわずしかも外部記憶等もほとんど使わずできる限り高速でコンパイルできるコンパイラを作成することはむずかしい．そこで，教育用（および研究用）として目的の異なる特別なコンパイラがこれまでいくつか開発されている．代表的なものとしては次のものがある．まず，FORTRAN では PUFFT (the Purdue University Fast Fortran Translator, パデュー大学, IBM 7090/94)[1], WATFOR (WATerloo FORtran, ウォータールー大学, IBM 7040/360)[2][3][4], DITRAN (DIagnostic forTRAN, ウィスコンシン大学, CDC 1604/UNIVAC 1100/Burroughs B 5500)[5][6], WATFIV ("the one after WATFOR" または WATerloo Fortran IV, ウォータールー大学, IBM 360/370)[3], FLAG (Fortran Load And Go, バンダービルド大学, XDS Σ 7/5 (MELCOM 7700/7500))[7], RALPH (メリーランド大学, UNIVAC 1100)[8] がある．また，COBOL では WATBOL (WATerloo co BOL, ウォータールー大学, IBM 360/370), PL/I では PL/C(コーネル大学, IBM 360/370), アセンブラでは ASSIST (Assembler System for Student Instruction and Systems Teaching, ペンシルバニア州立大学, IBM 360/370) がある．この他にも多少古くなるが，MAD (Michigan Algorithm Decoder, ミシガン大学), CORC (the CORnell Computing language, コーネル大学) などがある．

これらのほとんどのものは主記憶常駐型であり，したがって1パス・コンパイラである．本稿では，このうち，FORTRAN コンパイラに限って解説を行なう．

2. 特徴

主記憶常駐型 FORTRAN コンパイラの特徴はおよそ次のとおりである．

（1） コンパイル速度が高速である．
（2） システム・オーバーヘッドが小さい．
（3） コンパイル時および実行時に詳細な診断が行なわれる．
（4） 短いプログラムの処理が効率的に行なえる．
（5） 制御文が簡単である．
（6） 結果の印刷がコンパクトである．
（7） 1枚のカードに2つ以上の文をパンチできる．
（8） オブジェクト・デックはとれない．
（9） FORTRAN 語以外の言語との混用はできない．

通常のコンパイラを用いて1つのジョブをコンパイルし実行する際にかかる主な時間としては次のものがある．

（I） コンパイラ・ロード時間
（II） コンパイル時間
（III） コレクション時間
（IV） 実行可能プログラム・ロード時間
（V） 実行時間

これらのうち（I），（III）および（IV）はシステム・オーバーヘッドとみなすことができる．ここで，主記憶常駐型 FORTRAN コンパイラについて，これらの時間をみてみると，まず，コンパイラを主記憶に常駐させることにより（I）がなくなる．ただし，これは一括処理を前提としている（4. 参照）．ついで直接機

* 慶応義塾大学情報科学研究所

械語に翻訳し目的プログラムは主記憶に置くこと，FORTRAN 語以外の言語との混用を禁止すること，オブジェクト・デックを出力しないことにより（Ⅲ）および（Ⅳ）がなくなる．そして，多くの場合，使用者はコンパイル時の文法チェックに時間がかかり，無事文法上の誤りがなくなったあかつきには極く少量のデータを処理するだけなので，実行時間よりもコンパイル時間を短縮することが重要である．そこで，一般には，最適化を行なわないことによってコンパイル速度を速めている．

診断に関しては，プログラム作成者が未熟であることを前提に，コンパイル時はもちろん実行時にも綿密な診断を行ない，親切なメッセージを出すようになっている（5. 参照）．

最近の大型機ではハードウェアが高速かつ豊富になったこととからみ，機能的にも速度的にも教育用として十分役立つものもあるが，それでもなおこのような特徴を備えたコンパイラは依然として意義がある．

3. 言語仕様

言語仕様は一般に JIS 7000 を越えている．しかし，同一機種の最適化を行なうコンパイラと言語上の互換性を保つよう努力されている．

教育用として特に配慮されている点として書式の制御がある．FORTRAN 語では書式の制御が最もわかりにくいところのひとつであるにもかかわらず，最初からある程度のことは知らなければならない．そこで，すくなくとも最初のうちは FORMAT 文の詳細は知らなくてもすむように入出力が単純化されている．たとえば WATFOR/WATFIV を例にとると次のとおりである*．

入力 READ, X, N または READ (5, *) X, N というような文でデータを読込むことができる．対応するデータ・カード上の数値はコンマか空白かで区切る．必要に応じて1枚あるいは数枚のデータ・カードを読込むことができる．しかしカード上にデータが残った場合には，それらのデータは捨てられる．データの型は入力並びの型と一致していればよい．

出力 PRINT, X, N または WRITE (6, *) X, N というような文でデータを印刷することができる．整数型は I12 で，実数型は E16.7 で出力される．さらに出力並びには，式を書くこともできる．ただし，式は左括弧ではじまっていてはいけない．〔FLAG では，この形の文ではデータの個数は8という制約がある．その代り INPUT/OUTPUT 文がある．これは NAMELIST を使用した場合と似た効果がある．最近の大型機の FORTRAN でも FORMAT 文の書式仕様を空にすることによって標準書式を指定するものがあることはある[9]．〕

この他の言語仕様上の特色としては多重代入文，添字式としての一般の式などがある．WATFIV ではさらにデータの型として文字型がある．

4. 処理方法

使用者のジョブは一括してまとめ，その前にコンパイラを呼び出すための制御文をつけるのが一般的かつ効率的な使い方である．すなわち，通常のジョブはサブジョブ化し，システム的にはこのサブジョブの集合が1つのジョブとして扱われるようにするのである．したがって，この間コンパイラは主記憶に常駐し連続したサブジョブの流れを処理する．サブジョブを構成するものは1つ以上の FORTRAN 語のプログラム単位とデータであり，一般にはこれだけで1つの実行可能プログラムを構成していなければならない．サブジョブにはこの他にサブジョブの宣言文およびデータとの区切りのための制御文が必要であるが，プログラム単位間に制御文を必要としない．

原始プログラムは逐次機械語に翻訳され，目的プログラムは直接主記憶に置かれる．実行可能プログラムを構成するすべてのプログラム単位のコンパイルが終了すると，目的プログラムに制御が渡される．

典型的なコンパイルの方式としては WATFOR/WATFIV 方式と PUFFT/FLAG 方式とがある．まず，WATFOR/WATFIV では実行可能プログラムを構成するすべてのプログラム単位のコンパイルが終了すると，データ領域が割付けられ，プログラム単位間のつなぎがとられる．データ領域はすべて共通ブロックとしてまとめられる．共通ブロックとして宣言されていない変数や配列は“uncommon”共通ブロックとしてまとめられる．そして，生成された命令の番地部は，定数および一時記憶を指すものを除き，すべてこれらのデータ領域を指すようううめかえられる．実行時に誤りが発生したときには，その位置あるいはデータは原始プログラムにもとづいて指示される．そこで，コンパイルされた文，生成されたコード，およびコンパイル後に割付けられるデータ領域間のつなぎ役としての記号表が実行時にも保持されている．このた

* これらは list-directed 入出力文と呼ばれる．

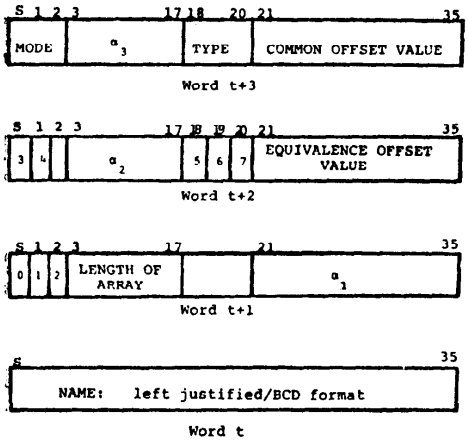

Fig. 1 Symbol table entry for a variable name (NAME) (文献[2]より転載)

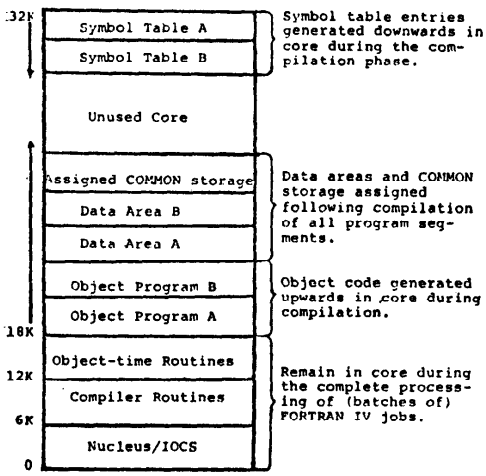

Fig. 2 Layout of core storage (K=1024, 36-bit words) (文献[2]より転載)

め，主記憶には番地の低い方から順次，命令コード，データ領域がとられ，記号表は番地の高い方から低い方に向って作られる．WATFOR の記号表および主記憶内のレイアウトを **Fig. 1** および **Fig. 2** に示す．

これに対し，PUFFT や FLAG では使用者プログラムのための領域がデータ領域とプログラム領域とにあらかじめわけられる．どこでわけるかはシステム生成時に定めるが，使用者がプログラムの性格に応じて制御文で指定することもできる．データ領域は，コンパイル時には記号表のために用いられ，実行時には変数，配列，定数，共通ブロックのデータ領域として用いられる．そこで，記号表は実行時には存在しない．それどころか，記号表の，外部名以外のプログラム単位に局所的な名前は，そのプログラム単位の翻訳が終わると破棄される．ただし，定数は外部名と同じに扱われる．プログラム領域は，宣言文の処理の際に作業用領域として使われた後，目的プログラム，コンパイル時に初期値が設定されるデータ，書式仕様，および NAMELIST 表を格納するために用いられる．式をコンパイルする際に使用する押込みリストもこの領域に作られる．コンパイル時にデータ領域を使いきってしまったときには，空いていればプログラム領域の高い番地がデータ領域として用いられる．PUFFT の主記憶内のレイアウトを **Fig. 3** に示す．

いずれの処理方式にせよ，数値変換，FORMAT 文の解読などを含む入出力のためのプログラム，診断プログラム，添字計算プログラムといった実行時プログラムおよび基本外部関数，組込み関数などはすべて主記憶に常駐する．

WATFOR コンパイラの主なルーチンおよびそれらの間の制御の流れを **Fig. 4**（次頁参照）に示す．

5. 診断機能

教育用主記憶常駐型 FORTRAN コンパイラは，主記憶常駐という制約があるにもかかわらず診断機能が豊富である．出される警告とか診断メッセージは，実習を行なっている学生にとってわかり易いものでなければならない．診断メッセージは，WATFOR/WA-

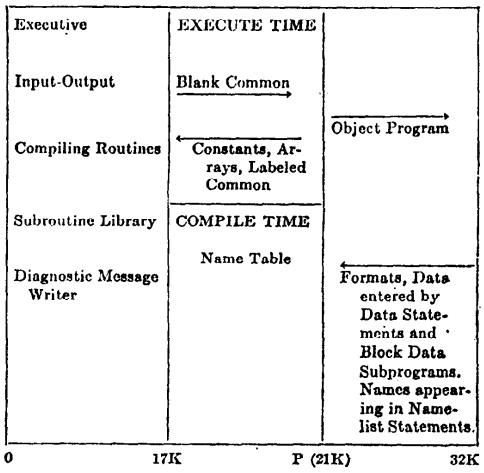

Fig. 3 Storage assignment in PUFFT (文献[1]より転載)

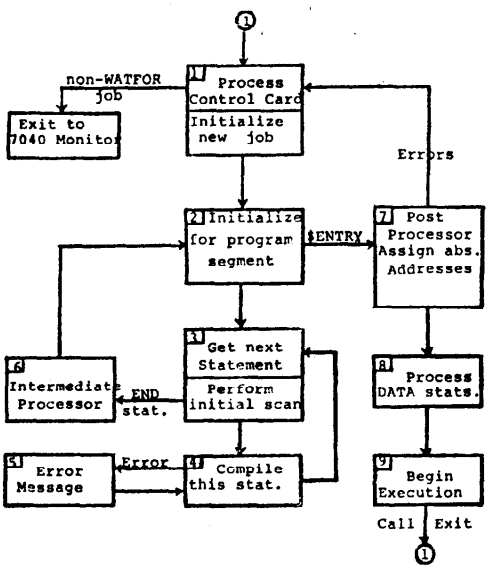

Fig. 4 Flow of control between main compiler routines (文献[2]より転載)

TFIV では原則としてコードで表示され，PUFFT/FLAG および DITRAN では英文で表示される．WATFOR/WATFIV, FLAG および DITRAN の診断メッセージの数はおよそ次のとおりである．

WATFOR	287
WATFIV	293
FLAG	201（単に情報を提供するだけのものも含む）
DITRAN	331

また，コンパイル時の診断メッセージは何段階かになっている．たとえば，WATFOR/WATFIV では拡張警告 (extension)，警告 (warning)，誤り (error) の3レベルある．拡張警告は FORTRAN IV を拡張した部分の使用に対するメッセージであり，他との互換性を考慮したものである（ただし，標準書式の使用に対しては何ら警告は出されない）．警告は仮定をたて解釈し先に進める誤りに対するもので，たとえば英字名が7文字以上の場合には警告を出し最初の6文字を採用する．誤りは実行に移ることができないものに

```
      $JOB,   WATFOR
      $ID             100010
      C     SS-1
1           DIMENSION X(5)
2           DO 1 I=1,10
3         1 X(I)=I
4           WRITE (6,100) X
5       100 FORMAT(1H ,5F10.5)
6           STOP
7           END
      $ENTRY

                    EXECUTION
      ERROR     SS-1  IN LINE 3      (STATEMENT     1 +     0 LINES)

           I                                6
      $IBSYS

COMPILE TIME          266     TOTAL TIME          416
OBJECT PROG            3R     DATA STORAGE          7     AVAILABLE CORE     11757     SYMBOL TABLE          32
```

(a)

```
      $JOB   WATFOR
      $ID             100010
      C       SR-5
1             DO 2 I = 1,5
2           2 CALL B(I)
3             STOP
4             END
5             SUBROUTINE B(I)
6               I=I + 1
7             RETURN
10            END
      $ENTRY


                    EXECUTION
      ERROR     SR-5  IN LINE 7      (STATEMENT     2 +     5 LINES)

               I

 SUBROUTINES IN REVERSE ORDER OF CALLING

   RCUTINE        CALLED FROM          BY ROUTINE
     B          LINE NUMBER 2           *MAIN*
      $IBSYS

COMPILE TIME          316     TOTAL TIME          483
OBJECT PROG            32     DATA STORAGE          3     AVAILABLE CORE     11757     SYMBOL TABLE          32
```

(b)

Fig. 5 Examples of WATFOR error messages

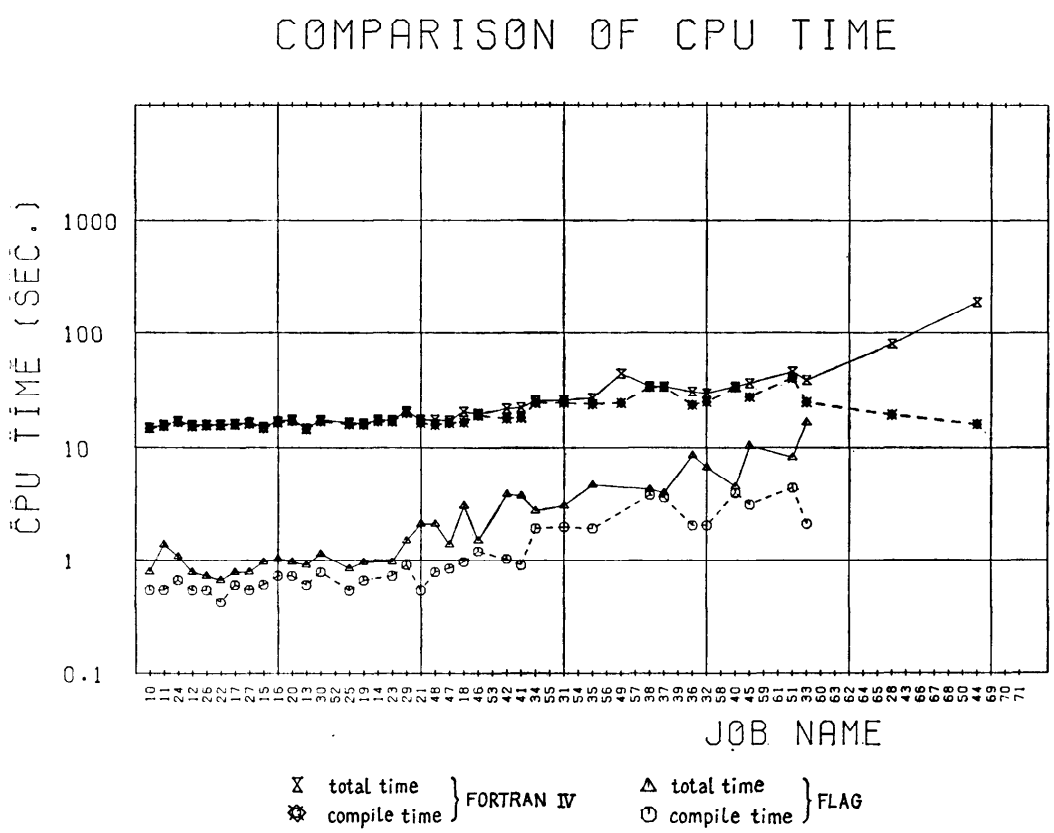

Fig. 6 The comparison of the processing time: FLAG vs. FORTRAN IV (used MELCOM 7700)

対するメッセージである.

実行時の診断メッセージは2段階ある. すなわち, 実行を直ちに中断する場合とそうでないものとに分けられる. たとえば, 桁あふれが生じると, 生じた時点でその旨のメッセージが出されるが, 何回か (WATFOR では 10 回) 生じるまではひき続き実行される.

実行時の診断機能は処理系によって, それぞれ特徴がある. WATFOR/WATFIV では原始プログラムの各文にコンパイラがつけた番号およびその文の直前の文の番号 (その文よりも前に文の番号がなければゼロ) とその文からの相対文数とで誤りのあった文が明示される (コンパイル時も同じ) だけでなく, さらに, 誤りを起したデータに対する英字名およびそのときの値が明示される. そのために, たとえば実行文に対する最初の目的コードが, コンパイラのつけた番号をレジスタに設定する命令になっており, 現在実行中の文につけた番号を常に保持している. WATFOR の診断メッセージの一例を **Fig. 5** (前頁参照) に示す.

PUFFT では, 原始プログラムの各文をリストする際に, その文に対する目的コードの先頭のコードの格納番地が8進数で表示される. これが原始プログラムと目的コードとの間のクロスリファレンスの役割をはたす. 実行時に誤りが生じると生じた番地が8進数で表示されるのである.

FLAG ではデバッグ・モード (オプション) を指定しない場合の診断機能はいささかさびしい. しかし, デバッグ・モードを指定すると, WATFOR/WATFIV の場合と同様に, コンパイラが各文につけた番号で誤りの生じた文が明示される. さらにデバッグ・モードでは, 添字の値のチェック, 副プログラムの引用の際の引数の型および個数が一致しているかどうかのチェック等ある程度の診断機能が目的プログラムに組込まれる (しかし, WATFOR/WATFIV にはおよばない). その結果, 目的プログラムの大きさは正常

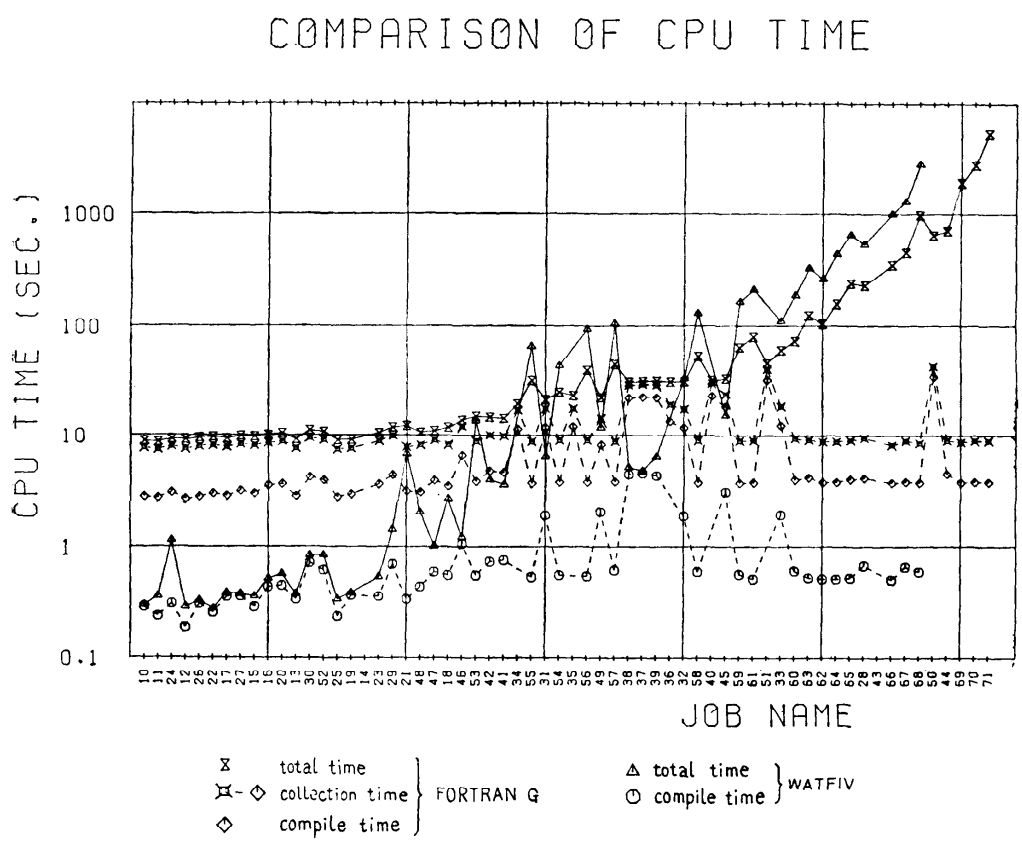

Fig. 7 The comparison of the processing time: WATFIV vs. FORTRAN G (used IBM 370/135 OS/VS1)

モードの場合とくらべて4割増位になる.（東京大学教育用計算機センターでは，オプション指定の省略時解釈がデバッグ・モードになっているようである.）

これに対し，DITRAN はその名が示すとおり，コンパイル時および実行時の誤りの検出および解析に特に力をいれており，論理的な誤り以外の誤りはすべて検出し原始プログラムにもとづいて指示することを売り物にしている．そのために，記憶単位の概念を拡張し，各記憶単位を制御部と情報部とから構成している，実行はインタープリティブ方式をとり，その結果，COMMON および EQUIVALENCE による結合の結果生じる定義，未定義のチェック，DO の範囲内における DO の制御変数および各パラメタの再定義のチェックなどきめのこまかい診断が行なわれる．さらに，パフォーマンスおよび使用に関するモニタリングも行ない，ジョブの終りに各サブジョブに対する情報を入手することができる．

未定義の変数あるいは配列要素の引用の検出機能としては 7040 WATFOR が最も秀れていた．すなわち，IBM 7040 では記憶単位に正常でないパリティをもった値を格納させることができ，しかもこれを引用したときには割出される．そこで Fig. 4 の 7 のポスト・プロセッサではデータ領域に番地を割付けるとともに，それらの場所にはパリティが正常でない値を格納する．こうすることによって，実行時に未定義の変数あるいは配列要素が自動的に検出されるのである．

6. 効率

すでに述べたとおり教育用主記憶常駐型 FORTRAN コンパイラはこまものを速くたくさん処理するのが得意なコンパイラで，しかもジョブの性質上コンパイル速度を速めることに重点が置かれている．そこでプログラムの大きさに関する種々の制約は一般の FORTRAN コンパイラよりも強い上に，実行時間が

```
      $JOB SAMPLE10  4110260785,KP=29
      C     EUCLIDEAN ALGORITHM (DIVISION FORM)
 1          INTEGER A, B, S, Q
      C     INPUT AND INITIALIZATION
 2          READ (5, 11) A, B
 3          L = A
 4          S = B
      C     TEST FOR ZERO REMAINDER
 5        3 IF (S .EQ. 0) GO TO 5
      C     PERFORM THE ALGORITHMIC CALCULATION
      C     RECALL THAT CHOP (L/S) WILL RESULT FROM
      C     THEFORTRAN INTEGER DIVISION, L/S.
 6          Q = L/S
 7          R = L - Q * S
 8          L = S
 9          S = R
10          GO TO 3
      C     PRINT THE GCD
11        5 WRITE (6, 12) A, B, L
12          STOP
13         11 FORMAT (2I15)
14         12 FORMAT (11H THE GCD OF, I15, 4H AND, I15,
             1  3H IS, I15)
15          END

      $ENTRY
THE GCD OF          87654 AND          24346 IS             14

CORE USAGE       OBJECT CODE=     576 BYTES,ARRAY AREA=       0 BYTES,TOTAL AREA AVAILABLE=   93800  BYTES

DIAGNOSTICS        NUMBER OF ERRORS=       0, NUMBER OF WARNINGS=       0, NUMBER OF EXTENSIONS=       0

COMPILE TIME=     0.37 SEC,EXECUTION TIME=     0.02 SEC,  WATFIV - VERSION 1 LEVEL 3 MARCH  1971      DATE=  74/134
```

(a) WAT

```
.JOE 002T9007,TS19
FLAG
  * FLAG VERSION B01          - 10119 JAN 19,'74 -
  * AVAILABLE MEMORY SIZE:
  *     PROGRAM & INITIALIZED VARIABLES = 3843 (WORDS)
  *     NON-INITIALIZED VARIABLES = 8954 (WORDS)
  *     TOTAL = 12797 (WORDS)

0001     C     EUCLIDEAN ALGORITHM (DIVISION FORM)
0002           INTEGER A, B, S, Q
0003     C     INPUT AND INITIALIZATION
0004           READ (5, 11) A, B
0005           L = A
0006           S = B
0007     C     TEST FOR ZERO REMAINDER
0008         3     IF (S .EQ. 0) GO TO 5
0009     C     PERFORM THE ALGORITHMIC CALCULATION
0010     C     RECALL THAT CHOP (L/S) WILL RESULT FROM
0011     C     THEFORTRAN INTEGER DIVISION, L/S.
0012           Q = L/S
0013           R = L - Q * S
0014           L = S
0015           S = R
0016           GO TO 3
0017     C     PRINT THE GCD
0018          5     WRITE (6, 12) A, B, L
0019                STOP
0020         11     FORMAT (2I15)
0021         12     FORMAT (11H THE GCD OF, I15, 4H AND, I15,
0022             1    3H IS, I15)
0023           END


  * ACTUAL PROGRAM SIZE:
  *     PROGRAM & INITIALIZED VARIABLES = 92 (WORDS)
  *     NON-INITIALIZED VARIABLES = 8 (WORDS)
  *     TOTAL = 100 (WORDS)


 THE GCD OF          87654 AND          24346 IS             14
*STOP*


  ** TOTAL TIME        .96        * COMPILE TIME      .66        * EXECUTE TIME      30
  ** * OF ERRORS    *NONE*        * * OF CARDS IN      27        * * OF PAGES OUT     2
```

(b) FLAG

Fig. 8

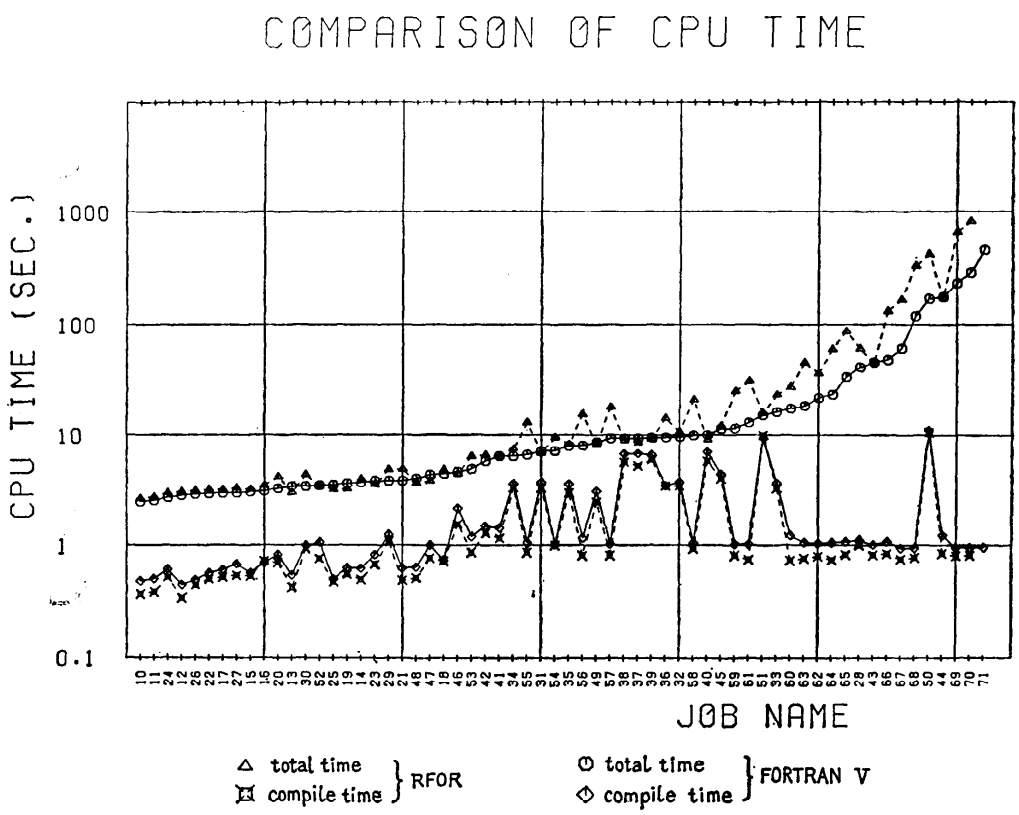

Fig. 9 The comparison of the processing time: RFOR vs. FORTRAN V (used UNIVAC 1106 (non-overlap memory))

長くかかるものの処理にはむいていない.

教育用主記憶常駐型コンパイラの処理効率をみてみるために, 東京大学教育用計算機センターの MELCOM 7700 の FLAG と FORTRAN IV および早稲田大学電子計算室の IBM 370/135 の WATFIV と FORTRN IV (G) にいくつかのプログラムをかけ, ジョブ当りの処理時間 (コンパイル開始から実行終了までに要する時間) を比較してみた結果が **Fig. 6** および **7** である (ジョブは UNIVAC 1106 FORTRAN V での処理時間に従って並べてある (**Fig. 9** 参照). 各ジョブの性質は **Table. 1** のとおり*). ただし, FLAG の方は実行時間が長いものが打切られているなどして特にデータが少ない. また, MELCOM 7700 の FORTRAN IV はコレクションに要した時間が不明である. ここで, 早稲田大学の IBM 370/135 は仮想記憶を採用しているが (主記憶は 240 KB), WATFIV および FORTRAN IV (G) はともに仮想記憶を考慮して作られたものでないということを考慮する必要があるかもしれない.

* これらのジョブは単に使用料金を比較するために作成したもので, コンパイラの特徴を解析するために作成したものではない.

7. 使用例

WATFIV および FLAG の使用例を **Fig. 8**(前頁参照) に示す. 印刷結果が極めてコンパクトにまとめられていることがわかる.

8. 問題点

主な問題点は2つある. その1つは誤りの取扱いとそれに対するメッセージに関する事柄である. まず, 教育用といっても入門的なプログラミング教育とその他の計算機教育とがあり, さらに研究用等にも利用す

Table. 1 各ジョブの概要

番号	プログラムの概要	
10	フィボナッチ数列	フィボナッチ数列の 1000 を越えた最初の項を求める.
11	フィボナッチ数列	1000 を越えた最初の項までの各項を求める.
12	給与計算	賃率と勤務時間数とを読込み給与を計算する. (データ・カード: 3 枚)
13	成績の集計	データを読込み, データの個数と平均値とを求める. (データ・カード: 4 枚)
14	判別式	データとして標題と 2 次方程式の係数を読込み判別式を計算する. (データ・カード: 5 枚)
15	最大値	N 個のデータを読込み最大値を求める. (データ・カード: 11 枚)
16	頻度	N 個のデータを読込み 3 つの組に分けて出現頻度を求める. (データ・カード: 12 枚)
17	ユークリッドのアルゴリズム	2 つのデータを読込み最大公約数を求める〔減算法〕.
18	ユークリッドのアルゴリズム	17 にトレース機能を加えたもの.
19	ユークリッドのアルゴリズム	2 つのデータを読込み最大公約数を求める〔除算法〕.
20	零和ゲーム	6×6 の配列を用いて零和ゲームを行なう. (データ・カード: 7 枚)
21	約数	$\sqrt{N}$ まで繰返し, N の約数をすべて求める. (N=654321098)
22	多項式	4 次式の係数と X の値を読込み, $E(X)$ を求める.
23	行列の各要素の和	$M\times N$ 行列の各要素の和を求める. (M=5, N=8. データ・カード: 11 枚)
24	入れ子になったDOループ	3 桁の数字で, それぞれの桁の 3 乗の和に等しいものをすべて求める.
25	倍精度の加算	倍精度実数型データを読込み, 和を求める. (データ・カード: 11 枚)
26	平方根	ニュートン法を用いて $\sqrt{A}$ を求める. (A=768.51238, ε=1.E−4)
27	sin x	x=0.3421, ε=1.E−7 を読込み sin x を求める.
28	積分	台形則を使って $f(x)=x^3+2.341$ を区間〔0.54243, 3.128〕で積分する. 誤差が 1.E−4 より小さくなるまで分割を細かくして繰返す.
29	積分	$f(x)=-x^3+2.43x+2.128$ を区間〔0.54243, 3.128〕で, 台形則, 中点則およびシンプソン則を使って積分する. 台形則の誤差の限界が 1.E−4 より小さくなるまで繰返す.
30	回帰直線	(x_i, y_i) を読込み, 回帰直線と標準誤差を求める. (i=4)
31	逆行列	サブルーチン副プログラムを使い N 次正方行列の逆行列を求める. (N=10)
32	逆行列	(N=20)
33	逆行列	(N=30)
34	逆行列	倍精度計算. (N=10)
35	逆行列	倍精度計算. (N=20)
36	逆行列	倍精度計算. (N=30)
37	多項式の根	2 項係数を係数とする N 次多項式の根をニュートン法で求める. (N=5)
38	多項式の根	(N=10)
39	多項式の根	(N=20)
40	多項式の根	倍精度計算. (N=10)
41	常微分方程式	$dy/dx=3y/(1+x)$ をルンゲ・クッタ・ギル法で解く. 初期値 x=0, y=1 で, x=10 まで 0.05 きざみで求める.
43	積分	$f(x)=e^x$ を区間〔0, 1〕でシンプソン則を使って積分する. 相対誤差が 1.E−4 以下になったら終了する.
44	積分	倍精度計算.
45	分散分析	5 因子の分散分析. (データ: 144 組)
46	N 元連立一次方程式	N 元連立一次方程式を掃出し法を使って解く. (N=3)
47	ヒストグラム	データを読込み, ヒストグラムを作成する. (データ: 44 個)
48	平方根	1 から 100 までの数の平方根をニュートン法で求める. 倍精度計算.
49	ファイル処理	親ファイルを作り, 報告書を作成してから, ファイルを更新し再び報告書を作成する. 1 枚のデータ・カードを 1 記録単位として扱い, 全部で 92 記録単位処理する.
50	偏微分方程式	境界条件 $\frac{1}{\xi^2}\left\{\left(\frac{\partial\psi}{\partial\xi}\right)^2+\left(\frac{\partial\psi}{\partial\eta}\right)^2\right\}=\kappa(1-\eta)$ のもとで $\frac{\partial^2\psi}{\partial\xi^2}+\frac{\partial^2\psi}{\partial\eta^2}-\frac{1}{\xi}\frac{\partial\psi}{\partial\xi}=0 \qquad \xi\neq 0$ を SOR 法で解く.
51	回帰分析	経済データの回帰分析. (データは 31 期分 90 枚)
52 ～ 71	積分	$f(x)=x^5+x^3-1.32512$ を区間〔0, 100〕でシンプソン則を使って積分する. 慶応義塾大学の UNIVAC 1106 システムで多重度 1 のとき FORTRAN V での実行時間が, ほぼ 0.4, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 45, 60, 120, 240, 300, 480 (秒) になるようにそれぞれきざみ幅を調整したものである. プログラムは非局所的な最適化がきかないようになっている.

べきなので，診断は何段階か段階を設け，使用者が必要に応じて指定できることが望ましい．また，誤りに対するメッセージは，一般に誤りの指示にとどまり，それを取り除く助けとなる情報がない．言語上の禁止事項に違反した場合，ものによっては何も検出せず実行し終了してしまう（たとえば，結合によって生じる未定義の値の引用など．結果は正しくないはず）．さらに実行時の入出力関係の誤りの場合には，多くの場合ふちょうもどきが印刷されるので，使用者にとっては何か誤りがあったらしいという程度のことしかわからない．こういった類の誤りに対する診断は，（教育用だから必要ないという意見もあるが）特に必要と思われる．ただし，1パスでしかもできるだけ実行時間も短くしようとするとむずかしいが，デバッキング・エイドの1つとして用意しておく必要があるだろう．そして，入門的なプログラミング教育の場合には，1つでも誤りがあった場合には，実行に移らないようにすべきであろう．

第2の問題は，多重プログラミング方式およびカフェテリア方式と常駐型との兼ね合いである．まず多重プログラミングの場合には必要とされる数だけコンパイラを主記憶に常駐させる必要がある．さらに，場合によっては，CPU 効率やターンアラウンド時間を考えると，ジョブの大きさ（サブジョブの数）などのバランスを考慮する必要がある．またカフェテリア方式をとる場合には，単一プログラミング，多重プログラミングにかかわらず，システム・オーバーヘッドを減少させるために各ジョブをサブジョブ化するといった常駐型のメリットを生かすことがむずかしい．特に端末が幾つもある場合には，現在のところでは制御プログラム自体を修正しなければならないことが生じるだろう．つまり各ジョブをサブジョブ化しない場合は，常駐型でなくなってしまう．こういったことから，RALPH (FORTRAN V に MAD を合わせた機能をもつ）は再入可能型コンパイラである．ただし，コンパイル・アンド・ゴー型ではない．現在，慶応義塾ではこの RALPH に制約を加えた RFOR (Reentrant FORtran) を用いている．〔慶応義塾では，この両者の利点をとった，再入可能型でしかもコンパイル・アンド・ゴー型の FORTRAN コンパイラおよび実習用アセンブラを現在開発中である．〕RFOR と FORTRAN V との処理時間を比較したものを参考のため **Fig. 9**（前頁参照）に示す．

9. むすび

WATFOR を中心にいろいろな角度から主記憶常駐型 FORTRAN コンパイラについて述べてみた．

わが国では教育施設には現在のところ比較的小型のシステムが設置されているようである．そのようなシステムでは，システムによっては常駐型のコンパイラの作成は不可能かもしれない．しかし，システム・オーバーヘッドの減少およびコンパイル速度の高速化を目的とした教育用のコンパイラは工夫すれば作成可能であろう．この種の例としては HITFOR[10] がある．これはインタープリタ方式をとっていて，しかも常駐型ではない．

コンパイル速度を高速化することによって，教育にとどまらず，プログラム作成段階のデバッギング・エイドのひとつとしても利用することができる．また，積極的に利用すべきである．そこで，システムには，最適化を行なうコンパイラとここで述べた類のコンパイラの2種類が備わっていて，しかも当然のことながら言語仕様上互換性があることが望ましい．計算機教育が広まるにつれて益々教育用コンパイラの必要が高まるであろう．

最後に，本稿をまとめるにあたり東京大学教育用計算機センターならびに筧捷彦（東京大学，現立教大学）氏および早稲田大学電子計算室ならびに宇都宮公訓（早稲田大学，現筑波大学）氏には計算機の使用から結果の集計まで大変お世話になった．また，当研究所の大野義夫，山本喜一の両氏にはデータを整理していただいた．これらの方々に，この紙面をかりて御礼を申し上げる次第である．

参 考 文 献

1) Rosen, S. et al.: "PUFFT-The Purdue University fast FORTRAN translator," Comm. ACM, Vol. 8, No. 11 (Nov. 1965), 661-666.
2) Shantz, P. W. et al.: "WATFOR-The University of Waterloo FORTRAN IV Compiler," Comm. ACM, Vol. 10, No. 1 (Jan. 1967) 41-44.
3) Cress, P. et al.: "FORTRAN IV with WATFOR and WATFIV," Prentice-Hall, Inc., Englewood Cliffs, New Jusey (1970).
4) 7040 FORTRAN IV SER GUIDE, Digital Computing Center, University of Waterloo (1965).

5) Moulton, P. G. et al.: "DITRAN-A Compiler Emphasizing Diagnostics," Comm. ACM, Vol. 10, No. 1 (Jan. 1967), 45-52.
6) DITRAN Reference Manual, Academic Computing Center, The University of Wisconsin-Madison (1972).
7) MELCOM 7000 FLAG 説明書, HM-SR00-24A〈35B0〉, 三菱電機株式会社 (1973).
8) Reid, B. K.: "A Guide to Programming with RALPH," The University of Maryland (1970)
9) UNIVAC 1100 シリーズ FORTRAN V 解説書, 2651059, 日本ユニバック株式会社(1974).
10) 金山 裕・他: "教育用 FORTRAN システム HITFOR", 第15回プログラミングシンポジウム報告集, 155-173 (1974).

(昭和49年8月10日受付)