FACOM 222 FAST† (CDT 402 型)*

辻ヶ堂　信** 丸　山　武**

1. 概　　説

1・1 序　　論

FACOM 222 FAST (CDT 402 型) は FACOM 222 用の FORTRAN 型コムパイラであり，コア 4,000 語，ドラムなし，磁気テープ2台，紙テープ読取機1台，紙テープ穿孔機1台およびラインプリンタ1台をもつ計算機を対象としている．その原始プログラムで使用可能な言語は，7090 FORTRANにだいたい似ているが，磁気テープ，磁気ドラムを制御する記述とequivalence は使えない．

このコムパイラ作成の基本方針としては

（1） 翻訳時間を短かくするために，原始プログラムの走査の回数をできるだけ少なくし，また中間言語としての symbolic assembler 段階を省略し，機械語にできるだけ近い相対形式のプログラムを作り出すこと．

（2） 複数のサブプログラムを全く別の時点にコムパイルしておき，まとめて遂行することができること．

（3） 簡単な計画に従って，コムパイルと遂行が任意に繰り返せること．

（4） 複素演算が行なえること．

（5） 翻訳過程では，Lukasiewicz 表現 (Polish prefix notation) を利用すること．

などが挙げられる．

コムパイルに要する時間は，1行の記述当り平均5秒であり，現在，各サブプログラム当り使用可能な被演算子の名称の数などに関し相当制限があるが，一部 handcoded program を用いて，CHAIN-JOB および磁気テープの読み書きを行ない，大規模な計算をされた方もある．

1・2 使用し得る記述と制限条項について

このコムパイラで使用し得る記述は次のようなものである．

数式；GO TO n；ASSIGN i TO N；GO TO N, (n_1, n_2, ……, n_m)；GO TO (n_1, n_2, ……, n_m), I；IF (expression) n_1, n_2, n_3；SENSE LIGHT i；IF (SENSE LIGHT i) n_1, n_2；IF (SENSE SWITCH i) n_1, n_2；IF OVERFLOW n_1, n_2；DO n I=m_1, m_2, m_3；CONTINUE；PAUSE n；STOP n；END (I_1, I_2, I_3, I_4, I_5)；FORMAT；PRINT n, 変数に関する情報；FEED i または FEED N；SKIP TO T 2 (or HOME) POSITION；READ PAPER TAPE n, 変数に関する情報；PUNCH PAPER TAPE n, 変数に関する情報；FUNCTION NAME (a_1, a_2, ……, a_n)；SUBROUTINE NAME (a_1, a_2, ……, a_n)；RETURN；CALL NAME (a_1, a_2, ……, a_n)；DIMENSION V (l, m, n), ……；COMMON A, B, C, ……；COMPLEX A, B, C, ……；F テープ；

このうち，COMPLEX という記述は，これに続く被演算子は複素変数であるという宣言であり，これにより一つの変数名称に対して2語の記憶装置が割り当てられるが，一般の被演算子と同様にアドレス計算を行なうため，実数部と虚数部を別々の領域にとることにした．なお宣言以外の複素被演算子を含む記述においては，その最初に K, という字を打っておかなくてはならないし，また IF 中の表現では複素演算は行なえない．

使用できるライブラリ関数は LOG, SIN, COS, EXP, SQRT, FLOAT, ATAN および TAN H の8種であり，複素演算においても使用できる．

ライブラリ関数および AS 関数 (Arithmetic Statement Function) の actual argument としては任意の表現つまり値をもち得るものしか許されないが，サブプログラムに対する actual argument としては，その他，ライブラリ関数の名称，この (サブ) プログラム中で定義された AS 関数の名称および F テープで指定された場合には，サブプログラムの名称を書くことができる．たとえば SNAME 1 と SNAME 2 とをサブルーチン・サブプログラムの名称とするとき

```
CALL SNAME 1 (A+B*C, SNAME 2, SIN,
    E, F)
```

のような書き方が許される．この actual argument

---

* On a FORTRAN-type compiler FACOM 222 FAST (CDT 402), by Makoto Tsujigado and Takeshi Maruyama (Fujitsu Limited)

** 富士通信機製造株式会社

† Fuji Automatic Sentence Translator の略称

に関する規約は複素演算に対しても適用される．

END (I_1, I_2, I_3, I_4, I_5) の各 I_i は，コンソール上の Alteration Switch に対応し，$I_i=0$ または1のときは実際の switch に優先し，$I_i=2$ のときは実際の off または on によるが，その機能は次のように規定されている．

ASW 1=on：コムパイル終了後直ちに目的プログラムの遂行が可能な状態になる．すなわち目的プログラム・ローデイング・ルーチーンが内部記憶装置中に磁気テープより読み込まれていったん休止する．始動釦を押すことによりローデイングが開始される．

ASW 1=off：コムパイル終了後再びコムパイルを行なうことができる状態でいったん休止する．

ASW 2：この on または off により，コムパイル中，プログラムの部分を印刷するかしないかの制御を行なう．

ASW 3=on：現在までにコムパイルされた（サブ）プログラムと次に入って来るサブプログラムにおいて，同一名称の被演算子は同一の対象を指すものとしてコムパイルする．これに対して off の場合には，同一名称の被演算子も各サブプログラムごとに全く別のものを指示するものとしてコムパイルを行なう．

ASW 4：この switch の on または off により，目的プログラムは紙テープに穿孔されるか，あるいはテープ穿孔機は全く動作しない．なお ASW 5 は常に off にしておく．

制限条項は主として記憶装置の大きさに起因するものであり，コムパイルの段階では各サブプログラムにつき，次のような制約が加わる．

（1） 記述の数として

DIMENSION+COMMON+COMPLEX+AS 関数+F テープ+このサブプログラム中で呼ぶサブプログラムの数≦9

もし，この大きさを越えると T OVFL 107 という表示がライン・プリンタに印刷される．

（2） 記述中の元の数として

3*(DIMENSION の元の数)+2*(COMMON の元の数)+1*(COMPLEX の元の数)+2*(AS 関数の定義式の数)+2*(このサブプログラム中で呼ばれる FS 関数の数)+2*(このサブプログラム中で呼ばれるサブルーチーン・サブプログラムの数)+1*(F カードで宣言された関数の数)≦29

もし，この表で溢れがおこると T OVFL 108 という表示がライン・プリンタに出る．たとえば

```
DIMENSION A(3, 5, 7), B(10, 12, 13)
COMMON A, E, F
DIMENSION C(100), D(90), G(3, 6, 10),
          H(10)
```

とある場合，DIMENSION の記述の数は二つであり，COMMON の記述の数は一つであり，また，ここでいう元の数とは，DIMENSION については A, B, C, D, G, H の六つ，COMMON については A, E, F の三つである．このようないわゆる宣言的な6種の情報に対しては，六つの表を作るより二つの表を用いて，読み込まれた順にすきまもなく重なり合いもないように並べる方が記憶装置の節約になると考えたので，上記（1）および（2）の制限が生じた．

（3） 定数または変数については

整定数の数≦14，浮動定数の数≦34，変数名称の数≦49，Dummy Argument の数≦5，記述番号の数+DO 記述の数≦50，Assigned GO TO の n_1 の数≦30

で，これを越えると各々溢れの表示が出る．

（4） 1行の記述に関する制限

1行の記述中の元の数はサブスクリプトを除いて67以下のこと．

サブスクリプトは1行の記述につき14元まで使用可能である．

一つの AS 関数の定義式中に現われる Dummy Argument の総数は20以下のこと．

（5） 一つのサブプログラム中に現われる Dummy Argument の数は，全部で10以下のこと．それ以上現われて来る場合は，このサブプログラムに附属する変数でおきかえておくこと．

これらの制限を越えると各々 Table Overflow の表示が出る．

Object Program に関する制限としては，

（1） FAST Objectprogram Loader と管理プログラムとで，内部記憶装置 2,000～3,999 を占めるゆえ，全プログラムと全定数を合わせて1,930語を越えることはできない．

（2） 複素演算ルーチーンが必要とされるときは内部記憶 1,400～2,229 を占める．このルーチーンが内部記憶中にはいるか否かは，管理プログラムが ASW 1 に尋ねてきめる．

（3） ライブラリおよび入出力制御ルーチーンは 2,230～3,870 を占める．

（4） Supervisor は 3,880～3,999 を占める．

2. コムパイラについて

2・1 全体系一覧（全体系一覧図参照のこと）

本コムパイラは

（1） 原始プログラムを読み込んで，必要な表を作りながら被演算子にはコードつき相対アドレスを対応させ，また演算因子をコード化しながら全情報を磁気テープ No. 2 上に書き込む部分（3,245 語）と

（2） この磁気テープ上の情報を走査して，Format 情報を output する部分（約 700 語）と

（3） この磁気テープ上の情報より目的プログラムを作り出す部分（2,775 語）と

（4） 定数や変数と相対アドレスとの対応表や，記述番号と目的プログラムの相対位置との対応表を output する部分（約 800 語）よりなる．

磁気テープ No. 1 には，このコムパイルが四つのブロックに分けられて記憶されており，必要に応じて順次内部記憶装置に運ばれる．

磁気テープ No. 2 はコード化された原始プログラムを記憶しておくものであり，（1）でコード化された情報は 60 語を 1 ブロックとしてまとめられて書き込まれる．（1）が終了すると各種の一覧表が書きこまれたのち rewind され，（2）の段階では Format 情報がこのテープより読み出され，紙テープに命令と同じ形式で output されていき，走査が終ると rewind される．（3）の段階では，この磁気テープの情報より目的プログラムが作成され，できたプログラムは直ちに紙テープに output されていく．（4）の段階では，もし ASW 3=on ならば，各種の一覧表が磁気テープ No. 2 より内部記憶中に移されたのち，rewind されて，一つの（サブ）プログラムのコムパイルが終了する．

2・2 数式の翻訳について（数式のコード化のブロック・ダイアグラム参照のこと）

コード化されている情報より機械語のプログラムを作成するに際して Lukasiewicz 表現にいったん変換し，ここから目的プログラムを作り出している．

Lukasiewicz 表現は次のように定義される．

（1） 被演算子は表現である．

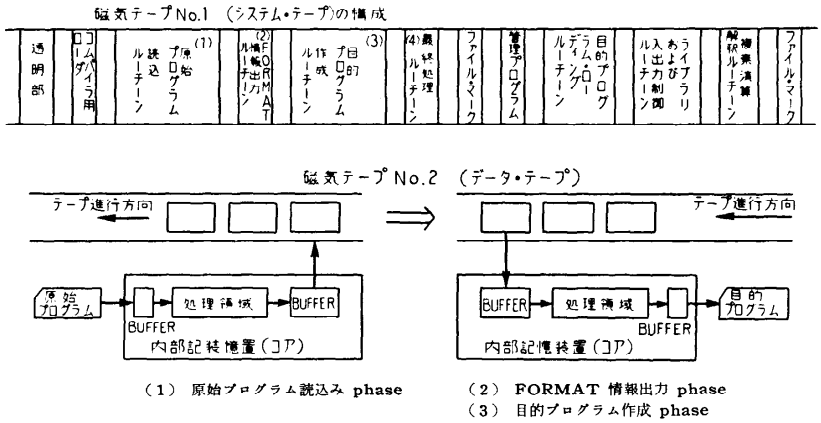
（2） $z_1, z_2, \cdots\cdots, z_n$ を表現とし，F_n を n 項に作用する演算因子とするとき，

$$F_n\, z_1\, z_2 \cdots\cdots z_n$$

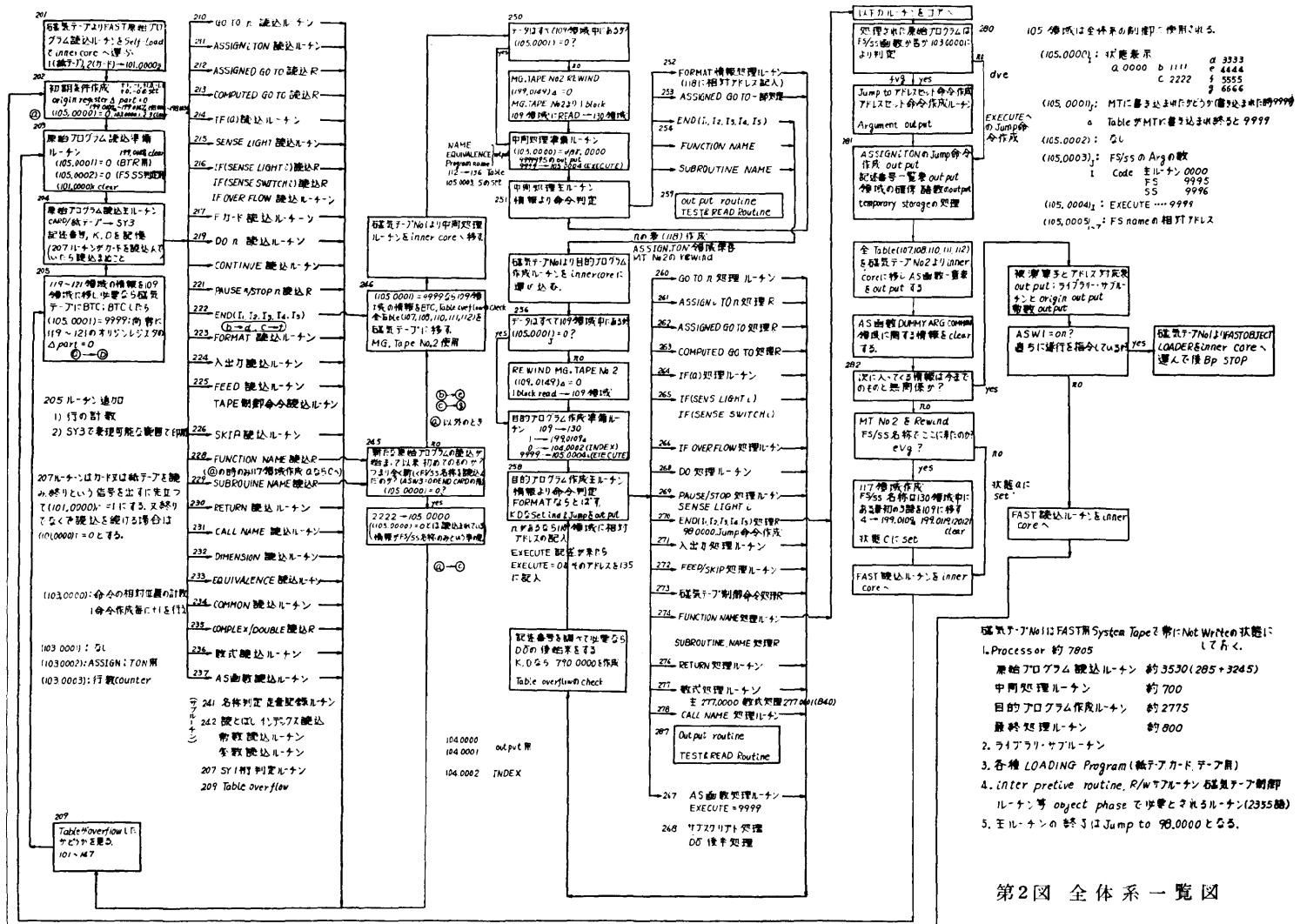
は表現である．

なお数式 変数＝表現 も 表現＝変数 と位置をかえて内部記憶におくときは，＝も他の 2 項演算因子と全く同じように，前にあるものが第一被演算子，後にあるものが第二被演算子と考えられるゆえ，数式と表現とを翻訳過程では区別しない．そしてこの Lukasiewicz 表現を逆に並べ，被演算子には 1 を，F_n には $1-n$ を割当てた数列を作り，左はしよりその数列の部分和を作っていくとき，この部分和がすべて 0 より大で，かつ最後が 1 ならば，この表現は正しい表現であるという定理を利用して表現の検査を行なっている．次にこの記号の列を左から右に走査し，演算因子に出会うたびに対応する被演算子を探して目的プログラムを作り出している．

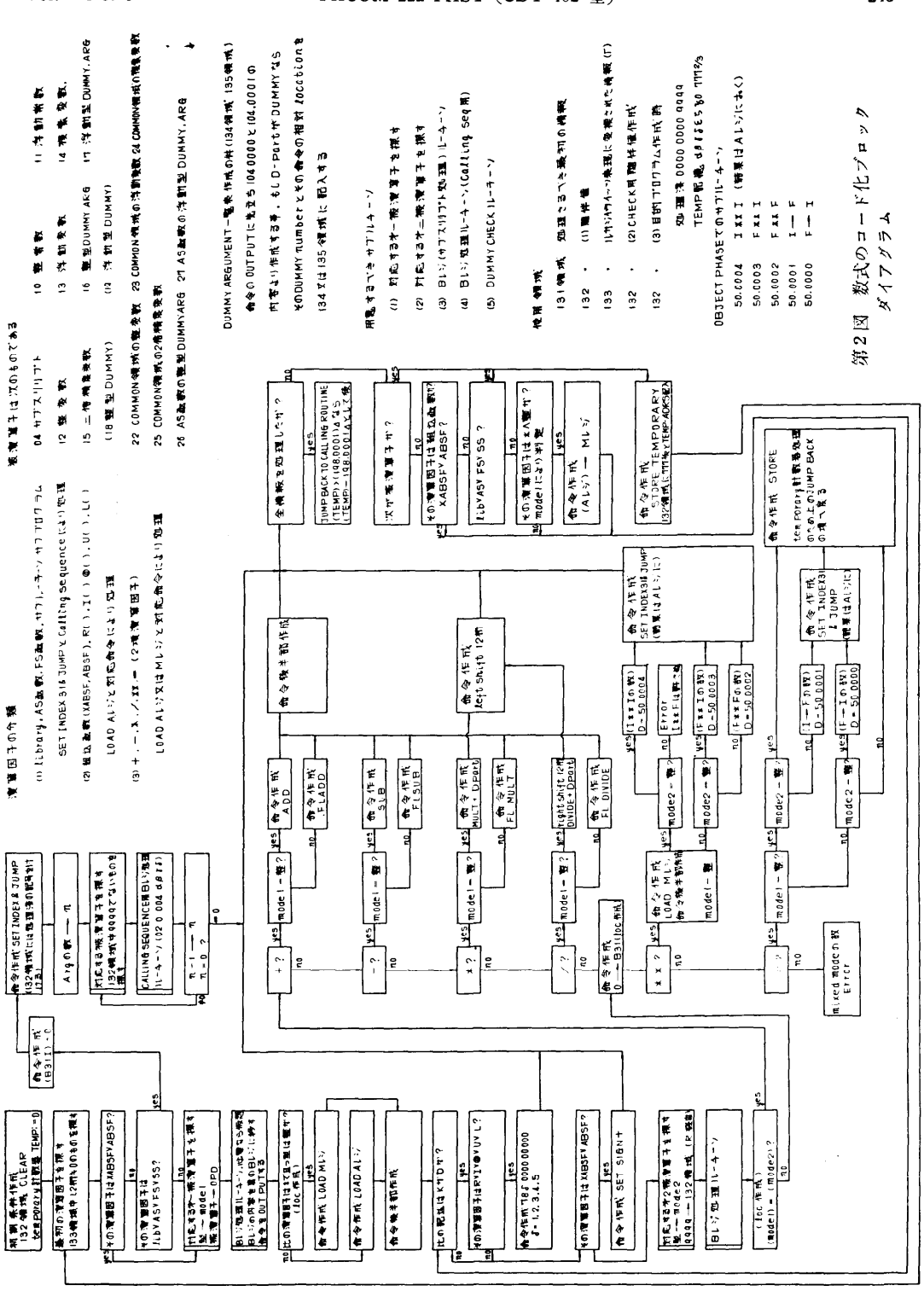
サブスクリプトについては，各記述の処理をはじめる前に処理してしまい，1 行ごとの optimization を



第 1 図　システム・テープの構成とデータ・テープ


201
磁気テープよりFAST原始プログラム読込ルーチンをSelf (load & inner) core へ運ぶ
1(鉄テープ), 2(カード)→101.0000
202
初期条件作成
origin register 3 part =0
(105.0001)=0 assume 1,3 tape
203
原始プログラム読込準備ルーチン
199 word clear
(105.0001)=0 (BTR用)
(105.0002)=0 (FS SS判定用)
(101.0000) clear
204
原始プログラム読込ルーチン
CARD/紙テープ→SY3
記述番号, K, D を記憶
(207ルーチンがカードを読込んでいたら読込まること
205
119~121 領域の情報を109領域に移し, 必要なら磁気テープにBTC; BTCしたら (105.0001)=9999; 向 前に119~121のオリジンレジスタ90 Δpart =0
Ⓔ→Ⓑ
205ルーチン追加
1) 行の計数
2) SY3で表現可能な範囲で印刷
207ルーチンはカードまたは紙テープを読み, 終りという信号を出すに先立って(101.0000)=1にする. 又終りでなく読込を続ける場合は(101.0000)=0とする.
(103.0000): 命令の相対位置の計数
1命令作成毎に+1を行う
(103.0001): なし
(103.0002): ASSIGN; TON用
(103.0003): 行数counter
209
Table overflow したかどうかを調る.
101~147
210 GO TO n 読込ルーチン
211 ASSIGN; TON 読込ルーチン
212 ASSIGNED GO TO 読込R
213 COMPUTED GO TO 読込R
214 IF(a) 読込ルーチン
215 SENSE LIGHT 読込ルーチン
216 IF(SENSE LIGHT) 読込R
IF(SENSE SWITCH) 読込R
IF OVER FLOW 読込ルーチン
217 Fカード読込ルーチン
219 DO n 読込ルーチン
CONTINUE 読込ルーチン
221 PAUSE/STOP n 読込R
222 END(I, I2, I3, I4, I5)
Ⓑ→α, C→Z
223 FORMAT 読込ルーチン
224 入出力読込ルーチン
225 FEED 読込ルーチン
TAPE制御命令読込ルーチン
226 SKIP 読込ルーチン
228 FUNCTION NAME 読込R
229 SUBROUTINE NAME読込R
(Ⓐの時のみ117領域作成 ならびに)
230 RETURN 読込ルーチン
231 CALL NAME 読込ルーチン
232 DIMENSION 読込ルーチン
233 EQUIVALENCE 読込ルーチン
234 COMMON 読込ルーチン
235 COMPLEX/DOUBLE 読込R
236 数式読込ルーチン
237 AS函数読込ルーチン
(サブルーチン)
241 名称判定 定義記憶ルーチン
242 読とばし インデクス読込
前数読込ルーチン
後数読込ルーチン
207 SY1用 判定ルーチン
209 Table overflow
245
前たる原始プログラムの読込ヲ始ます, 2以後 初めのCardのオプションより今と前(FS/SSも名称も読込みたのち (ASW3=ONEND CARDの時) (105.0000)=0?
yes
2222→105.0000
(105.0000)=0とは読込まれていた情報がFS/SSを終わるかどうか判定
Ⓐ→Ⓒ
246
(105.0001)=9999なら109領域の情報をBTC, Table overflow Check 全Table(107, 108, 110, 111, 112)を磁気テープに移す.
MG, Tape No.2 使用
磁気テープNoより中間処理ルーチンをinner core へ移す
NAME EQUIVALENCE (object Program name) 112→136 Table 105.0003 Set
250
データはすべて(109領域中にあるか (105.0001)=0?
yes
MG, TAPE No2 REWIND (109, 0149)a =0
MG, TAPE No2より1 block 109領域にREAD→130領域
中間処理準備ルーチン
(105.0000)=dat, 0000 9999→95 on out put
2222→105 SOURCE (EXECUTE)
中間処理ルーチン
(105.0001)=0 assemble
251
情報より命令判定
252
FORMAT情報処理ルーチン (118に相対アドレス記入)
253
ASSIGNED GO TO 部処理
256
END(I, I2, I3, I4, I5)
FUNCTION NAME
SUBROUTINE NAME
259
out put routine
TEST & READ routine
n の番 (118) 作成
ASSIGN TON 領域番号
MT No2 の rewind
磁気テープNoより目的プログラム作成ルーチンをinner core に運び込む.
256
データはすべて109領域中にあるか (105.0001)=0?
yes
no
REWIND MG, TAPE No 2 (109, 0149)a =0
1 block read → 109領域
目的プログラム作成準備ルーチン 109→130
1→199, 0107a
0→104.0002 (INDEX)
9999→105.0004 (EXECUTE)
258
目的プログラム作成主ルーチン
情報より命令判定
FORMATならとばす.
KDS Set Line & Jump & out put
nがあるGOSUB領域に相対アドレスの配入
EXECUTE 記述が来たらEXECUTE=04 そのアドレスを135に記入
記述番号を調べて必要なDOの修飾子をする
K, Dなら 790 0000を作成
Table overflow の check
260 GO TO n 処理ルーチン
261 ASSIGN TO n 処理R
262 ASSIGNED GO TO 処理R
263 COMPUTED GO TO 処理R
264 IF(a) 処理ルーチン
265 IF(SEVS LIGHT i)
IF(SENSE SWITCH i)
266 IF OVER FLOW 処理ルーチン
268 DO 処理ルーチン
269 PAUSE/STOP 処理ルーチン
SENSE LIGHT i
270 END(I, I2, I3, I4, I5) 処理R
90.0000 Jump 命令作成
271 入出力処理ルーチン
272 FEED/SKIP 処理ルーチン
273 磁気テープ制御命令処理R
274 FUNCTION NAME 処理ルーチン
SUBROUTINE NAME 処理R
276 RETURN 処理ルーチン
277 数式処理ルーチン
主 277.0000 数式処理 277.0001 (B40)
278 CALL NAME 処理ルーチン
287
Output routine
TEST & READ Routine
247 AS函数処理ルーチン
EXECUTE=9999
268 サブスクリプト処理
DO 修飾処理
104.0000 out put 用
104.0001
104.0002 INDEX
以下カルーチンを コアへ
280
処理された原始プログラムは FS/SS 函数が各々 103.6000 により判定
no
dve
yes
Jump to アドレスセット命令作成 アドレスセット命令作成ルーチン
Argument out put
EXECUTE への Jump命令作成
281
ASSIGN TON の Jump命令作成 out put
記述番号一覧表 out put
領域の種々 諸数 out put
temporary storage の処理
全 Table(107, 108, 110, 111, 112) を磁気テープNo2 より inner core に移し AS函数一覧表 & out put する
AS函数 DUMMY ARG (common 領域に関する情報を clear する.
282
次に入ってくる情報は今までのものと無関係か?
yes
no
MT No 2 を Rewind FS/SS 名称をここに書かのか? eVg?
yes
no
117領域作成 FS/SS 名称は130領域中にある最初の3語を109に移す 4→199, 0104, 199, 0105, 0202 clear
状態Cに set
FAST 読込ルーチンを inner core へ
105 領域は全体系の制御に使用される.
(105.0000)1: 状態表示
a 0000 b 1111 c 2222 d 3333 e 4444 f 5555 g 6666
(105.0001)1: MTに書き込まれたかどうか(書き込まれ時9999
a Table が MTに書き込まれ終ると 9999
(105.0002): なし
(105.0003)1: FS/SS の Arg の数
I Code 主ルーチン 0000
FS 9995
SS 9996
(105.0004)1: EXECUTE …… 9999
(105.0005)1~7: FS name の相対アドレス
被御要素とアドレス対応表 out put: ライブラリー・サブルーチンと origin out put 前数 out put
ASW1=on? 直ちに操作を指令してみる
yes
磁気テープNoよりFASTOBJECT LOADERをInner Core へ運んで後 Bp STOP
no
状態aに set
FAST 読込ルーチンを inner core へ
磁気テープNo1はFAST用System Tapeで常にNot Writeの状態にしておく.
1. Processor 約 7805
原始プログラム読込ルーチン 約 3530 (285+3245)
中間処理ルーチン 約 700
目的プログラム作成ルーチン 約 2775
最終処理ルーチン 約 800
2. ライブラリー・サブルーチン
3. 各種 LOADING Program (紙テープ, カード, テープ用)
4. inter pretive routine, R/W サブルーチン 磁気テープ制御ルーチン等 object phase で必要とされるルーチン (2355語)
5. 主ルーチンの終りは Jump to 90.0000 となる.

第2図 全体系一覧図



第2図 数式のコード化ブロック ダイアグラム

行なっている.

なおコムパイラ作成中論理的におこりえないと思われる約100個所において error 表示を行なっている. 原始プログラム読み込みの部分で 69, Format 情報 output 段階で二つ, 目的プログラムを作成する部分で 21, その他七つである. error が生じた時には直ちにその行の処理を中止し, 次の行の処理を行なうことにより, できるだけ計算機を停めることのないようにしている.

3. 目的プログラムと管理プログラムについて

3・1 相対形式の目的プログラムについて

一つの (サブ) プログラムは変換されて, この (サブ) プログラムの名称, この (サブ) プログラム中で呼んでいる他のサブプログラムの名称とコード, 機械語に近い相対形式の命令の集合, 記述番号と相対アドレス対応表, 一時記憶領域を指定する情報, 各領域の大きさに関する情報, AS 関数と相対アドレスとの対応表, 使用したライブラリ一覧表, 定数に関する情報の順に紙テープに output され, さらに目的プログラムのローダーに対する制御情報として, コムパイル時の ASW 3 に従い, いったん休止後以下のプログラムを現在までのものと被演算子は同じオリジンをもつものとして計算機械中に読み込め, または被演算子に関してすきまもなく重なりあいもないようオリジンをずらして読み込めというコードと, これで目的プログラムは終了の故 execute の phase で必要な各種ルーチーンを内部記憶に移して, MAIN プログラムの第1アドレスに制御を移す用意をしたのち, いったん休止せよというコードが穿孔される. ライン・プリンタにはこのほか原始プログラム中で使用された変数名称と相対アドレスとの対応表が印刷される. なお, このプログラム中, IF 型および GO TO 型の記述に対応する jump 命令と, サブプログラムを呼ぶ記述に対応する jump 命令と, 一時記憶を被演算子とする命令には浮動アドレスの記号がつけられており, ローディングのとき, 芋づる制御により真のアドレスがうめられる.

3・2 FAST Objectprogram Loader について

この Loader は一つの Mainprogram と全く別の時点でコムパイルされた複数個のサブプログラムを内部記憶中に入れることと, hand-coded program も続けて Load 可能なことと複素演算解釈ルーチーンのため絶対形式となった命令コードにいろいろな記号をつけることを目的とし, ローディング終了と共に消されて, その代りにライブラリーや入出力制御ルーチーンが書かれる.

3・3 入出力関係制御ルーチーンについて

FAST コムパイラは入出力関係の記述を jump 命令と calling sequence に変換し, それ以上のことは execute phase において入出力関係制御ルーチーンが, Format 情報を解釈しつつ被演算子に対応させ, input または output を行なっている. Format 情報中機械を動作させるコードは, 入力機械に対しては (と／であり, 出力機械に対しては／と) である.

3・4 サブプログラムとその補助ルーチーンについて

サブプログラムは次のような構造を execute phase でもつ.

(1) JUMP TO (6)
(2) FORMAT 情報
(3) ASSIGNED GO TO に対応する jump 命令
(4) AS 関数
(5) EXECUTE 部
(6) ACTUAL ARGUMENT を DUMMY ARGUMENT に対応させるルーチーンへの jump 命令
(7) DUMMY ARGUMENT および RETURN に対応する命令のある LOCATION に関する情報
(8) JUMP TO (5)
(9) 一時記憶用領域

コントロールはサブプログラムの頭より (6) を経由して actual argument を dummy argument に対応させるルーチーンに jump する. このルーチーンは Calling ルーチーン中の Calling sequence と (7) の情報を参照して dummy Argument のある命令のアドレス部を, 対応する actual argument のアドレス部で置換していき, 最後に RETURN に対応する命令のアドレス部に所定のアドレスを植えて(8)へ戻る. ここから制御は EXECUTE 部に移り, やがて RETURN に対応する命令を経由して Calling ルーチーンに戻っていく. この actual argment を dummy argument に対応させるルーチーンは約 50 語よりなり, 入出力制御ルーチーンと同様 execute の段階では常に内部記憶中におかれている.

3・5 Hand-coded Program との結合

FACOM 222 はコア記憶装置の中 01～99 番をインデクス・レジスタとして使用し得るが, この中 01～29

番はプライオリティ用に使用するため，普通 30 番から使用する．

そこで FACOM 222 FAST ではサブスクリプトのそのときの値を入れてアドレス変更を行なうために B 30 を使用し，サブルーチーンとの結合に B 31 を使用し，dummy argument に actual argument を対応させるために B 32 を使用している．また入出力制御ルーチーンは B 33～B 39 を使用している．これらの中 B 30 は Calling ルーチーンに戻って来るたびに回復されるゆえ，結合のためには B 31 を使うことという制約のほか，hand-coded program ではインデクスレジスタの使用については制約をもたない．ただし FAST で書かれているサブプログラムから戻って来たとき，B 30～B 39 は一般には変っている．

上記の注意の許で hand-code を行ない，その頭にこのルーチーン名称と，このルーチーン中で呼んでいる他のサブプログラム名称とそのコードを附着し，コムパイラが output するのと同じ形式で紙テープ上に穿孔しておくことにより，他の目的プログラムに続いてローディングが行なわれて，他のプログラムと結合される．

4. FAST 複素演算解釈ルーチーン

4・1 複素数導入上の一般的問題

FAST システムに複素演算を導入する際におこる問題に処す基本態度は「コムパイラの負担を最小にする」というものであった．

まず，複素数の内部表現については，実部虚部の対 (x, y) のほかに，一応，より一般的な絶対値偏角の対 (r, φ) も考えられるが，実数の表現とのつながり，また倍精度演算との共通性という面から，前者をとる．この記憶装置内の配置についても（1）連続する 2 語をとり，おのおの実部虚部を割付ける，のと（2）実部群，虚部群とを分離して対応する実部虚部が一定番地差をもつようにする，と二た通り考えられるが，前者はアドレス計算が実数の場合と異なる．コムパイラの負担を最小にするという立場から後者を試みた．また複素数は浮動型に限定した．

LOG, EXP, SIN, COS 等ライブラリ関数には，もちろん，実数の場合と違う一般的定義を与え，偏角（の主値）を求める ARG を追加する．組込関数 ABS は，やや複雑になってライブラリ関数として扱う．右辺の表現を複素数の実，虚部に付値するものとして R( )= および I( )= をとり，任意表現の実部，虚部または共役をとるものとして R( ), I( ), @( ) も扱いうるようにした．これを用いて，$i^2=-1$ なる i を UI として，複素定数 $x+iy$ は $x+$UI$*y$ の形の表現で代用できる．演算時間はともかく直感的でもあるので，直接複素定数を与える記述形式はとらないこととした．

入出力に関しては，複素数を連続する 2 実数とみて Format には実部虚部の形式指定（必ずしも一致しなくともよい）を並べる．

4・2 解釈ルーチーン

コムパイラの負担を最小のものにするという立場から，実行段階において複素数演算は解釈ルーチーンが解釈実行する．複素数として用いる変数は COMPLEX と，宣言され，このように complex 宣言された変数を被演算数としてもつ算術式命令，入出力命令などはカードにあっては第 1 欄に K と，紙テープにあっては K, と打たれることで，K-カード命令であることが示される．コムパイラは K-カード命令を翻訳するに際し，まず解釈ルーチーンに入る命令を発生し，次で複素演算固有の演算子以外は意識せず普通に翻訳し，命令の終了を告げる疑似命令で結ぶ．

このとき，K-カード命令を翻訳したものの中にはアドレス計算のための整型演算が含まれるが，ローディング段階でローダがこれを検出し，真の機械語であるとして解釈ルーチーンが飛び越すべき範囲指示を挿入することで切り抜ける．

解釈ルーチーンの構造原理は，かなものの模倣であり幾つかの擬似レジスタをもつ簡単なものであるが，雑多な家事切りまわしがあり，ライブラリ関数などもふくめ，約 700 機械語を要した．

算術関数，関数サブプログラム，サブルーチーン・サブプログラムなども実数と同様に使用できるが，サブプログラムの actual argument として，K-カード命令が参照するような算術関数名称，サブプログラム名称は，ある種の構造上の欠陥から，現時点では用いられない．

4・3 複素演算のプログラム例

```
C       1/Z O GENTEN NO MAWARI NI
C       SEKIBUN.
        COMPLEXZ, DELTAZ, SQINT, UI
K       R(UI)=0.0
K       I(UI)=1.0
K       SQINT=0.0
K       Z=1.0
```

```
K       DELTAZ=−0.01+UI*0.01
        Dϕ 100 I=1,4
        Dϕ 10 J=1,100
K       SQINT=SQINT+DELTAZ/Z
K    10 Z=Z+DELTAZ
K   100 DELTAZ=DELTAZ*UI
K       PRINT 1000, SQINT
  1000  FϕRMAT (16 HSQUARE INTEGRAL
        =2 F 15.5)
        END (2,2,2,2,2)
```

第3行目は Z, DELTAZ, SQINT, UI なる変数が複素数であることを宣言し，第8行目は $-0.01+0.01i$ を DELTAZ に付値している．

5. 管理プログラムについて

一つの（サブ）プログラムのコムパイル終了後，さらにコムパイルを続けるか，または execute を行なうかは ASW 1 の off または on により指示されたのに対し，execute phase では内部記憶 3,880～3,999 上に Supervisor と呼ばれるプログラムがあり，特にデータ領域をこれにあててこわさぬ限り，loading と execute の間保持される．この Supervisor の機能は

（1）ローダを読み込み，制御をローダに移すこと

（2）ローディングするプログラムの途中に Supervisory 情報があれば，制御はいったんローダから Supervisor に移り，必要な処理を行なったのち，再びローダに制御を移すこと．

（3）Objectprogram 上の情報，BP-HALT & EXECUTE（コードとしては 9999998）により制御はローダから Supervisor へ移る．Supervisor はすでに記憶してある制御情報に従って System-tape を rewind し，あるいは Compile-Phase 用プログラムと Object phase 用プログラムとを区切る File-mark まで System-tape を戻して，コムパイラまたはローダを内部記憶に読み込む用意をしたのち，実行すべきプログラムの第1アドレスに jump をしていったん休止する．原始プログラムの主ルーチーン中の END 記述はこの Supervisor への jump 命令に変換されるゆえ，計算終了後制御が END 記述に行くように原始プログラムを書いておけば，execute 後自動的に次の compile または execute が行なわれる．

Supervisory 情報としては

（1） HLTCOM: Halt & Compile

いったん休止後始動ボタンを押せばコムパイルを開始するもので，Supervisory 情報を特に与えなければ Supervisor はこの制御を行なうもの．

（2） HLTLOD: Halt & Load

いったん休止後，始動ボタンを押せば，ローディングを開始するもの．

の二つがあり，この情報は内部記憶 3,942 におかれるが，新たな Supervisory 情報により更新される．ただし一度コムパイルを行なうと無効になり，標準型 HLTCOM に戻る．

おわりに

このコムパイラは現在東京工大計算機室において，Main Program Method としてご使用頂いています．コムパイラの誤りをいろいろ指摘くださいました東工大の方々およびこの FAST をさらに発展させ，磁気テープの使用と CHAIN-JOB を行なわれた小林先生に御礼申し上げます．また，いろいろご指導くださいました東大井上先生，富士通小林，松原，池田，山崎の諸氏およびいろいろご協力くださった今野，小松，村田，南，宮原，荻原，加藤，渡辺の諸君に感謝致します．

（昭和38年11月7日受付，同12月12日再受付）