FACOM-231 のための Assembler Language VALSAS Ⅳ

吉岡　敏彦*・佐々木幹生**・佐藤　文雄***

Revised Report on the Assembler Language for FACOM-231 VALSAS Ⅳ

Toshihiko YOSHIOKA, Mikio SASAKI and Fumio SATŌ

Abstract

In this report authors relate of **VALSAS IV**, Assembler for **FACOM—231**. The representative characteristics of it are One Pass system and Utility subroutines built into it. One Pass system : When the source program passes through the tape-reader one time, it is translated into the object program at once. The mean processing time is 0.5 second per one step. Utility subroutines are divided into two kinds as follows :

1. **Standard functions**

SQRT : 221/SQRT, A, B. **SIN** : 221/SIN, A, B. **COS** : 221/COS, A, B. **EXP** : 221/EXP, A, B. **LOG** : 221/LOG, A, B. **ANGTG** : 221/ANGTG, A, B. **ABS** : 221/ABS, A, B.

For example　221/SQRT, A, B.

〈221〉 is Link Order. **A** and **B** are location symbols which indicate the least significant digits of fields. When this order is executed,the value of square root of 〈**A**〉 is stored in **B** field.

2. **I/O subroutines**

READINTEGER : 221/READINTEGER, I. **READREAL** : 221/READREAL, A.

PRNTINTEGER : 221/PRNTINTEGER, I. **PRNT** : 221/PRNT, A.

PRNTREAL : 221/PRNTREAL, A, i. **PRNTFIX** : 221/PRNTFIX, A, i, j.

PRNTX : 221/PRNTX, I, i. **PRNTSTRING** : 221/PRNTSTRING, LS.

SPACE : 221/SPACE, i. **CRLF** : 221/CRLF, i.

These orders read in, print numerics or characters with various types.

CONTENTS

*電気工学教室
**日本電信電話公社電気通信研究所
***アルプス電気株式会社

1. まえがき

電子計算機にある問題を処理させようとするならば，それに先立ちその問題の解法手順を細かく指示するプログラムが必要である．そしてその手順を計算機に与え実行させるわけであるが，その手順をどのような言語で計算機に与えるかが問題になる．計算機言語は大きく分けると**機械言語** (Machine Language)，**アッセンブラ言語** (Assembler Language，または Symbolic Language ともいう)，**コンパイラー言語** (Compiler Language，代表的なものにALGOL，FORTRAN，COBOLがある) になる．

これらは計算機の進歩と共に開発されてきたもので計算機が広く普及し利用されるようになつたのもこれら計算機言語の進歩によりプログラムを組むことが非常に容易になつたためといつてもよい．

現在もつとも使い易いコンパイラー言語はそのすぐれた多くの点を持ちながらもどのようなプログラムも組めるというわけではない．特殊な情報処理を必要とする問題になると無力となる．コンパイラー言語で組めない問題の処理にはどうしても機械言語またはアッセンブラ言語の助けを必要とする．アッセンブラ言語は機械言語の特性を完全に内包し，プログラムの組み易さにおいても機械言語をはるかに凌いでいる．コンパイラー言語の欠点を補うばかりか真に user が計算機を自分の手で使いこなそうとするならばアッセンブラ言語の使用が最もよいと思われる．

FACOM—231 はその特殊な機構のため機械言語でプログラムを組むことは非常に困難を伴い，アッセンブラ言語の必要性は切実なものであつた．

その困難を克服するために我々は **FACOM—231** のためのアッセンブラ言語として **VALSAS IV** (Variable Word Length Symbolic Assembler System) を作製した．VALSAS は昭和38年より I，II，III と改良され，今日のIVに到つた．そして我々はあらゆる program に対して実験し，テストをくり返したが充分実用に耐えることを確認した．ここに自信を持つて報告する．

2. VALSAS IV の概要

1) **VALSAS IV** を使用する場合の所要機器は次の通りである．

- FACOM—231計算機本体 1式
  磁心記憶装置 32768桁
- 紙テープ読取装置 1台
- タイプライター (IBM I/O) 1台
- 磁気テープ装置 (機番3) 1台

2) 入出力命令として **PRNT, PRNTREAL, PRNTINTEGER, PRNTFIX, PRNTX, PRNTSTRING, SPACE, CRLF** が使える．

3) 標準函数として **SQRT, EXP, LOG, SIN, COS, ANGTG, ABS,** 換算函数には **FIX, FLOAT** が自由にプログラムの中で使える．

4) **One pass** 方式であるので assemble完了までに要する時間が短い．

5) source program 中の **VALSAS IV** の文法に反する誤りを1回の読み込みで殆んど検出する．

6) 浮動小数点演算を pseudo order **FLA, FLS, FLM, FLD** を使つて通常の計算と同様に処理できる．

7) variant を含む operation code を symbolic に自由に定義できる．

8) 未定義の symbolic address に relative step を許す．ただしその relative step の絶対値は $(125)_{10}$ 以下でなければならない．

9) **VALSAS IV** によつて machine language になおされた object program を希望する番地から格納することができる．この場合 program をいくつかに分けて格納することも可能である．

10) 異なつた symbol を250種類まで同時に処理できる．また program で不必要となつた symbol を自由に cancel できるので使用可能 symbol 数は実質的には250種類より多い．

11) 同一の location に location symbol を多重に定義できる．

12) program 中に自由に comment を入れることができる．

13) 磁気テープ中に前述の入出力，標準函数，換算函数および **VALSAS IV** の system program が格納してある．

14) **VALSAS IV** の system program の占有領域は 4200 番地より 7 xv 2 番地である．
object program の使用可能領域は 100 番地より 41xx番地である．

3. 命令の構成

3.1 命令の形式

FACOM—231 の命令語の標準形は各桁16進表示で11桁よりなる．

$$OPVA_4A_3A_2A_1B_4B_3B_2B_1$$

OP：命令の種類の指定

V：OP部で指定された命令について更に細かい指定を行なう．

$A_4A_3A_2A_1$：一般に演算数が格納される領域の指定

$B_4B_3B_2B_1$：一般に被演算数が格納される領域の指定

FACOM—231 は可変長語方式の計算機であるので命令語によつては標準形の11桁以上，あるいは以下の桁数の命令語もある．例えば次のように命令語の桁数の長さがいろいろ違うものもある．

| O |

| O P |

| O P V | $A_4A_3A_2A_1$ |

| O P V | $A_4A_3A_2A_1$ | $B_4B_3B_2B_1$ | $C_nC_{n-1}\cdots C_2C_1$ |

いま OPVを<OP>，$A_4A_3A_2A_1$を<A>，$B_4B_3B_2B_1$を<B>とおくと **VALSAS IV** の命令の一般形は　<**OP**>/<**A**>，<**B**>.

すなわちvariantと A addressの間にslashを，番地部の区切りにcommaを，命令の終りにperiodを用いる．

各部の syntax は以下のように定義する（<A>，<B>は共に<L>と書く）．

<OP>:＝<Machine code>｜<Mnemonic code>

<L>:＝<Direct address>｜<Indirect address>

<Direct address>:＝<Actual address>｜<Symbolic address>

<Actual address>:＝<Decimal number>｜=<Hexadecimal number>

<Symbolic address>:＝<Location symbol>｜*｜<Symbolic address>±<Decimal number>

<Location symbol>:＝<Alphabet>｜<Location symbol><Alphabet>｜<Location symbol><Decimal number>

<Alphabet>:＝A｜B｜C｜…｜Z｜a｜b｜c｜…｜y｜z｜

<Indirect address>:＝(<Direct address>)

3.2 Actual address

5桁以内の10進数，または‘=’が先行する4桁の16進数よりなる．10進数は **VALSAS IV** によりすべて16進数に変換される．したがつて16進数に変換しないでそのままobject programに格納する場合‘=’を前に置く．‘=’が先行する数字はそのまま object program の address 部に格納されるが，その数字は必ず4桁でなければならない．

例　A/13000，=03sx.

↓　(A=410)

41032u803sx

3.3 Symbolic address

symbol は（*を除く）EXIT. までに location symbol として一度必ず定義されておらねばならない．symbolがlocation symbolとして現われる以前に address として現われた場合（未定義 address と名付ける）も location symbol として現われた後にaddressとして現われた場合も（定義address と名付ける）無関係に対応のactual addressによりおきかえられる．ただし未定義 address の場合 relative step は±125の範囲内でなければならない．relative address (relative step を持つ address) の場合symbolのactual addressに relative stepを加えたものがその真のaddress となる．*はその属する命令の最上位，すなわち **OP** 部の第1桁の番地を指す．

例　A/DATA，*.

S/AREA−5，*+10.

↓　(A=410，S=420)

410123x0100

42001xt0115

但し　DATA=123x番地

AREA=0200番地

と定義され $(0100)_{16}$ から格納されるとする．

3.4 Indirect address

Actual address，symbolic address を左右括弧（ ）でかこめばよい．括弧内のaddressの対応する番地に $(8000)_{16}$ が加えられて格納される．

例　A/(100)，(DATA+5).

S/(*−10)，(=0500).

↓　(A=410，S=420)

41080649005

42081018500

但し　DATA=1000番地と定義され $(0100)_{16}$ より格納されるとする．

3.5 そ の 他

(1) 命令の一般形は<**OP**>/<**A**>，<**B**>. であるが

短縮形 <OP>/<A>. <OP>/.

拡大形 <OP>/<A>, <P_1>, <P_2>, <P_3>, …, <P_n>.

を許す.

(2) 1命令全体が machine language の場合はseparatorである／は必要でなく，命令の終りにperiod を打てばよい.

例
.34801000750.
.343.
.6.
.221750000010304.
(Word Mark はなくてもよい)

(3) 以下に示す命令では上に述べた方法と異なる例外的なaddressを使用する.

◦ **入出力命令および外部記憶命令**

A address には mode 指定がいるためいままでの A address 指定方法とは多少異なる. B addressに関してはいままでの指定方法と全く同じである. A address として direct address を使うとき(indirect addressの場合は一般の命令と同じである)次の注意が必要である. A_4桁だけにmode指定がある命令（紙テープおよびタイプライター関係の入出力命令，磁気テープおよび磁気ドラム関係の外部記憶命令）ではmode（machine languageと同じ）と桁数指定（10進数で指定）を別々に書くか，または'='つきの16進数 address を書くか，いずれかで A address を記述しなければならない. A_3桁にmode指定を持つ命令（ラインプリンター関係およびカード関係の入出力命令，磁気テープコントロール命令）では16進数addressのみが許される.

例 RMN/413, INPUT.
↓ (RMN=750)
.750400v2000
但し INPUT=2000番地

◦ **Flip Flop 関係命令**

operation code **26, 2w, 66**ではB address にFF指定（Flip Flop 指定）が必要であるが，FF名として下表のようなsymbolあるいは16進数 actual addressが許される. B address に indirect address を用いる場合は一般の命令と変りない.

例 FFB/=2200, LG.
↓ (FFB=260)
.26022001100

actual	symbol	actual	symbol	actual	symbol
0100	OVF	6100	ERIO	4200	LPEA
1100	LG	7100	ERL	5200	LPEB
2100	EQ	0200	EFMT	6200	EFCR
3100	SM	1200	ESMT	7200	EFCP
4100	EFPT	2200	ETMT		
5100	ALM	3200	LPMT		

◦ **Link 命令**

OPV=221の命令ではparameterとして任意の長さの数字を持ちたいときがある. このときはその数値の前に'↑'を打つ.

例 221/SUB, PARA 1, PARA 2,
↑123456, 100.
↓
.2212000100010201234560064
但し SUB=2000番地
PARA 1 =1000番地
PARA 2 =1020番地
に定義されているとする.

4. Control Order の種類

control orderは assembler に指示を与えるものであって，object program では実行されない. 以下14種のcontrol orderを **VALSAS IV** では使用する.

4.1 OPDEF.

mnemonic operation code を定義するとき最初につける.

例 **OPDEF.**
A=410 S=420 MOVE=341
HALT=062.

4.2 ENTER.

編集開始命令であって program の先頭につける.

4.3 EXIT.

編集完了命令であつて program の最後につける. EXIT. の代りに 20字以内の任意の <comment> を書いてもよい. 編集が完了すると **'exit'** あるいは <**comment**> を印字する.

program は **ENTER.** で始まり **EXIT.** で終る.

4.4 FL=<ℓ>.

<ℓ> ::=<Decimal number>

但し　$1 \leqq \ell \leqq 40$

4.10 で述べる control order **C : DFC/<FC>.** で与えられる浮動小数点定数の仮数部の長さを ℓ 桁とする. すなわち浮動小数点演算を ℓ 桁で行なわせたいときにその桁数を指定する control order である. これは**C : DFC/<FC>.** に先立ち与えなければならない. このcontrol orderが与えられないときは標準形式の浮動小数点定数（仮数部の長さ10桁）と見なして演算を行なう.

4.5 C : DA/<ℓ>.

<ℓ> ::=<Decimal number>

領域が ℓ 桁確保される.

例　**C : DA/5.**

↓

00000

4.6 C : DAW/<ℓ>.

<ℓ> ::=<Decimal number>

領域が ℓ 桁確保され, その最上位桁に Word Mark が設定される.

例　**C : DAW/7.**

↓

0000000

C : DA/<ℓ>. とのちがいは最上位桁に Word Mark が設定されることである.

4.7 C : DC/<c>.

<c> ::=<Unsigned number> | <Unsigned number>+ | <Unsigned number>−

<Unsigned number> ::=<Decimal number> | <Hexadecimal number>

定数 c が確保される. 符号桁が＋であれば最下位に偶数が格納され, −であれば奇数が格納される.

なお定数 c の任意桁に Word Mark がついておればobject programでも同じ位置にWord Markが設定される.

例　**C : DC/12345−.**

↓

123451

最下位桁の 1 はsign digitでここでは−であるので奇数の 1 となる.

例　**C : DC/5678+.**

↓

56780

最下位桁の 0 は sign digit でここでは＋であるので偶数の 0 となる.

例　**C : DC/1234567890.**

↓

1234567890

4.8 C : DCW/<c>.

定数cの最上位桁にWord Markが強制的に設定される以外は **C : DC/<c>.** と全く同じである.

例　**C : DCW/1234−.**

↓

12341

4.9 C : DLS/<LS>.

<LS> ::=<period 以外の letter string>

letter string が確保される. 最上位桁に強制的にWord Mark が設定される.

例　**C : DLS/X+Y=5.**

↓

6720683t35

例　**C : DLS/ANSWER.**

↓

415562664559

4.10 C : DFC/<FC>.

<FC> ::=<eemm…ms>

ee ::=指数部

mm…m ::=仮数部

s ::=sign 符号（＋の場合は省略してよい）

浮動小数点定数 <FC> が（ℓ＋3）桁の領域（4.4 参照）内に address の小さい方から順に格納され, sign digitは領域の最下位に設定される. また最上位桁にはWord Markが強制的に設定される. mm…m の桁数が ℓ より小さいときは（ℓ＋3）桁の領域内で mm…m の下位には 0 が格納され, ℓ より大きいとき（ℓ＋1）桁以下切捨てられる.

ここでは浮動小数点定数 $+0.\mathrm{mm}\cdots\mathrm{m}\times 10^{N}$ は eemm…m＋ (ee＝N＋50) として表現される. したがつて

$-50 \leqq N \leqq 49$ ($0 \leqq ee \leqq 99$) の範囲の数字を扱う．なお<FC>は必ずnormalize（正規化）されていなければならない．すなわち $0.1 \leqq 0.mm\cdots m < 1$
また0は指数部，仮数部，符号桁共に0として表わす．もう少しく詳しく例で説明すると

$-123.45 \longrightarrow$ 5312345−

$0.01234 \longrightarrow$ 491234+

53，49は指数部，12345，1234は仮数部を，＋，−は符号を表わす．

例
```
ENTER.
FL=15.
C : DFC/5025−.
        ↓          (−0.25を表わす)
5025000000000000001
```
として格納される．最下位桁の1は−であるから奇数の1が入る．

例
```
C : DFC/0.
        ↓
0000000000000
```
この場合は標準形式（FL=10）とし格納されている．指数部，仮数部，符号桁共に0となる．

例
```
C : DFC/53123.
        ↓          (123.0を表わす)
5312300000000
```

4.11 C : ORG/<L_0>;

<L_0>::=<Decimal number>|<Location symbol>+<Decimal number>|<Location symbol>−<Decimal number>

object program の格納開始番地を<L_0>とするcontrol orderである．<L_0>がLocation symbolで出現に先立ち定義されていなければその時の location counter の内容に1を加えたものが格納開始番地となる．すなわちこの場合見かけ上はorigin setというよりも location symbol の設定が行なわれたと同じ効果である．
programの最初にこの control order がなければ自動的に格納開始番地が $(0100)_{16}$番地にsetされ $(0100)_{16}$番地から格納されてゆく．

例
```
C : ORG/=1000.
A/=1050,  =2000.
S/=1500,  =1800.
    ・・・・
    ・・・・
```
A/=1050…以後の program を $(1000)_{16}$番地より格納することになる．

4.12 C : END/<L_0>,

<L_0>::=<Decimal number>|=<Hexadecimal number>|<Location symbol>|<Location symbol>+<Decimal number>|<Location symbol>−<Decimal number>

object program の実行開始番地を<L_0>とするcontrol order である．特に<L_0>にlocation symbol が用いられるときは先立つて定義されていなければならない．またこのcontrol orderはprogram中に何度も使われることがあるが最後に現われたものが有効である．この control order を読み込むとassemble を一時停止するので C : END/0. の形でassemble 一時停止命令として使用することもできる．

例
```
ENTER.
A/=1000, 1050.
S/=1060, 1080.
    ・・・・
C : END/=0100.
EXIT.
```
この program は $(0100)_{16}$番地より格納され $(0100)_{16}$番地より実行される．

4.13 C : EQU/<L_0>, <L_1>, …<L_n>.

<L_0>::=<Decimal number>|=<Hexadecimal number>|<Location symbol>|<Location symbol>+<Decimal number>|>Location symbol>−<Decimal number>
<L_i>::=<Location symbol> i=1, 2, …n
location symbol <L_i> (i=1, 2, …n) が<L_0>に対応する番地に定義される．<L_0>がlocation symbol の場合は出現に先立ち定義されていなければならない．

例 C : EQU/14^00, DATA1, DATA2.
DATA1, DATA2が36t0番地に定義される．
(14000は10進数なので16進数に変換される36t0となる)

例 C : EQU/=1t00, A, B, C.
A, B, Cが1t00番地に定義される．

4.14 C : CAN/<L_1>, <L_2>, …<L_n>.

<L_i>::=<Location symbol> i=1, 2, …n
location symbol<L_i>をcancel する control orderである．
<L_i>は出現に先立ち定義されなければならない．この control order は program 上不要となつたsymbol を cancel し，使用可能な symbol の数を増

加させる場合に用いると有利である．この処理が行なわれるとcancelしたsymbolと，それに定義されていたactual addressがタイプライターで印字される．

例 C：CAN/START，DATA１，DATA２．

但し　START＝2000番地
　　DATA１＝02st 番地
　　DATA２＝4000番地

と定義されているとする．

このときlocation START，DATA１，DATA２の定義がcancelされる．そしてcancelされた以後これらのsymbolは別の任意の番地に定義することができる．

cancelされると次のようにタイプライターでcancelされたsymbolとそのactual addressを印字する．

cancel
start　2000
data１　02st
data２　4000

4.15 (comment).

これはcontrol orderではないがprogram中にcommentを入れたい場合には左右括弧でくくつてやればその間にどのようなcommentを書いてもよい．(ただしperiodを書いてはいけない)．これが現われると読み飛ばされobject programには何んの影響も与えない．

例　(NEXT CALCULATION IS A+B−C).

5. Location Symbolの設定

location symbolはalphabetと数字の５字以内の組合せ（ただし第１字が数字であつてはいけない）を使用できる．20文字までは許されるが，その場合最初の５文字が有効となり６文字以後の文字は読み捨てられる．異なつたsymbolは**250種類**まで同時に使用できる．

location symbolの設定には以下の２種類の方法がある．

(１)<Location symbol>)

以下に続く命令，定数，領域の最上位桁のaddressにこのlocation symbolを定義する．入出力命令で扱われる定数，あるいは分岐先の設定に用いると便利である．

例 **MAIN)** 348/1，100.

↓

34800010064

MAINはこの命令の最上位桁（３の桁）のaddressに定義される．

(２)<Location symbol>〕

以下に続く命令，定数，領域の最下位桁のaddressにこのlocation symbolを定義する．

dataの演算，転送，あるいは命令のaddress部の修飾に用いると便利である．

例 **DATA1〕** C：DCW/1234+.

↓

12340

DATA１はこの定数の最下位桁（符号桁＋）のaddressに定義される．

VALSAS IVではlocation symbolの多重定義，すなわち同一番地に対し異なったsymbolを重ねて設定することもできる

例 **A) B) C〕** C：DCW/1234+．A，Bは定数の最上位桁（１の桁）に定義され，Cは定数の最下位桁（符号桁＋）に定義される．

しかし次のような定義は許されない．

例 **A〕 B〕** C：DCW/123.
　　A〕 B) C：DCW/123.

すなわち<Location symbol>〕のみの多重定義，および<Location symbol>)の前に<Location symbol>〕を定義することはできない．

6. Mnemonic Operation Codeの定義

VALSAS IVではoperation code２桁とvariant１桁を含め一つのmnemonic codeに定義することができる．mnemonic codeとしてはalphabetと数字の４文字以内の組合せで自由に定義できる．（ただし第１字は数字であつてはならない）

また**FLA, FLS, FLM, FLD**をmnemonic codeとして使用してはならない．これらはfloating pseudo orderといつて**VALSAS IV**では特別のcodeとして使用する．

mnemonic codeは100種類まで定義でき同一のmachine codeに対して多重定義は許されない．source program中に用いるmnemonic codeはprogramに先立つて以下の例のように定義しなければならない．

OPDEF．のつぎにmnemonic codeとそれに対応するmachine codeを‘=’で結んで列挙し，最後にperiodを打つ．machine codeとして通常**‘OPV’**の順の16進数３桁であるが，Vの不要（すなわちvariant不要）の命令では‘０OP’と表わす．

例 **OPDEF**.

ADD=410 SUB=420

MOVE=340 HALT=062.

mnemonic operation code の定義を誤りなく処理し終ると **'opdef ok restart'** とタイプライターが印字し停止する.

次にコンソールの「**PROCEED**」ボタンを押すと **ENTER**. で始まる program 本体の読み込みに移る.

7. 浮動小数点演算

FLA, FLS, FLM, FLDの pseudo order を使うことにより計算機本体が浮動小数点演算の機能を持っているかのように program を作成することができる.

浮動小数点四則演算命令の形式と, その機能を以下に示す.

FLA/<A>, <B>.

(<B>)+(<A>)⟶(<B>)

FLS/<A>, <B>.

(<B>)−(<A>)⟶(<B>)

FLM/<A>, <B>.

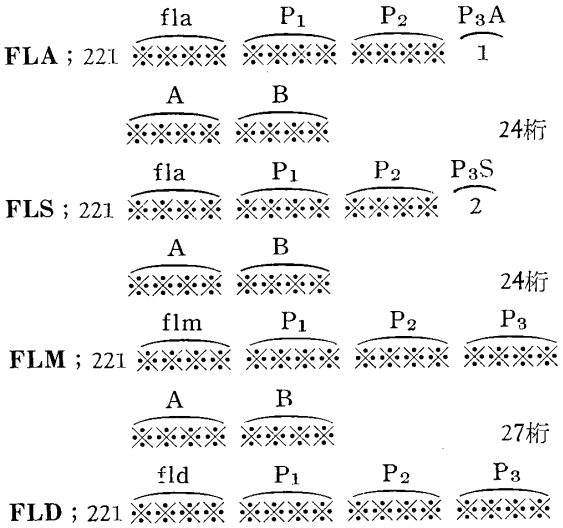
(<B>)×(<A>)⟶(<B>)

FLD/<A>, <B>.

(<B>)÷(<A>)⟶(<B>)

A address<A>, B address<B>の syntax は通常の命令 (3.) と変りない, A field, B fieldの数値は浮動小数点とみなされ, 両者の field の長さは必ず一致していなければならない.

これらの命令は**VALSAS IV**により次に示すような link命令に変換される.



ここに**fla, flm, fld**はそれぞれ**floating add and subtract, floating multiply, floating divid** の subroutineの最初の番地, A, Bはそれぞれ<A>, <B>の actual address を示す.

$P_1=PAD+\ell+2$

$P_2=PBD+\ell+2$

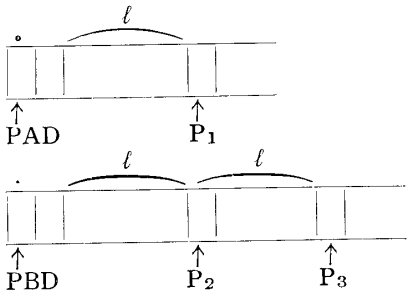
$P_3=PBD+2\ell+2$

ただしPADは43桁の作業領域の最上位桁の番地

PBDは84桁の作業領域の最上位桁の番地

ℓはFL=<ℓ>.により定義された浮動小数点定数の仮数部の長さを表わす.

これらの関係を図に示すとつぎのようになる.



P_1, P_2, P_3 のparameterはA field の数値を図の上の領域に, B field の数値を図の下の領域に移すために必要である. P_3A, P_3S は **floating add** と **floating subtract** の区別を示すための parameterである. なお浮動小数点演算中に overflow が生じる (すなわち10^{50}より大きい結果を得る) 場合には **'error 9'** を印字し, 演算を中止する.

8. Utility subroutines

8.1 入出力命令

8.1.1 READINTEGER

data を整数型で読み込む subroutine である.

使用法 221/**READINTEGER**, A.

A は定数領域の foot digit を指す Location symbol である.

符号と数字以外の文字は読みとばすが data の終りには必ず space か CRLF を少くとも一つせん孔する必要がある. data の桁数が定数領域の桁数より大きいときは data の上位の桁が切り捨てられる.

例 DATA」C:DAW/4.

のとき 221/**READINTEGER**, DATA.

が次のような data を読み込むと

data	DATA に読み込まれたときの形
12	0120
−123	1231
12345	3450
+6	0060
0	0000

となる．（最下位桁は符号桁を示し＋のときは0に，−のときは1になる）

8.1.2 READREAL

data を実数型で読み込む subroutine である．

使用法 221/**READREAL**, A.

Aは浮動小数点定数領域のfoot digit（符号桁）を指す location symbol である．

符号，数字，$_{10}$，period 以外は読みとばすがdataの終りには必ずspace，CRLFを少なくとも一つせん孔する必要がある．data の桁数が定数領域の桁数より大きいときは dataの下位の桁が切り捨てられる．

例 FL＝5.

A〕C：DFC/0.

のとき 221/**READREAL**, A.

がつぎのような data を読み込むと

data	Aに読み込まれたときの形
12	52120000
−12.567	52125671
0.003	48300000
$-15.678_{10}5$	57156781
$2.367_{10}-2$	49236700
−1234567	57123451
$0.0231_{10}-10$	39231000

となる．（最下位桁は符号桁で＋のとき0，−のとき1になる）

8.2 出力命令

8.2.1 PRNT

浮動小数点定数を実数型で印字する subroutine である．

使用法 221/**PRNT**, A.

Aは浮動小数点定数領域の foot digit を指す．

例 A〕C：DFC/561234567890+.

（123456.7890を示す）

このとき 221/**PRNT**, A.

が実行されると

$+1.234567890_{10}+05$␣

の形で印字される．印字されるとspaceが一つとられる．（␣はspace の意味である）

例 FL＝5.

DATA〕C：DFC/52352−.

のとき 221/**PRNT**, DATA.

が実行されると

$-3.5200_{10}+01$␣

と印字される．

8.2.2 PRNTREAL

使用法 221/**PRNTREAL**, A, i.

Aは浮動小数点定数領域の foot digit を指す．

i は印字する桁数を指示する．

すなわち実数型定数Aをi桁印字する subroutine である．

例 FL＝5.

DATA〕C：DFC/51462−.

このとき 221/**PRNTREAL**, DATA, 2.

が実行されると

$-4.6_{10}+00$␣

と印字する．印字の後space を一つとる．

PRNT と **PRNTREAL** のちがいは**PRNT**は浮動小数点定数領域の全桁数を印字するのに対し**PRNTREAL** は指定桁数だけ印字する．印字の様式はどちらも同じである．

8.2.3 PRNTFIX

使用法 221/**PRNTFIX**, A, i, j.

Aは浮動小数点定数領域のfoot digitを指す．

i，jは印字する桁数を指示するがiは浮動小数点定数の整数部の桁数を，j は小数部の桁数を指示する．そして印字の際整数部と小数部の間に小数点を印字する．

例 DATA〕C：DFC/5237524−.

のとき 221/**PRNTFIX**, DATA, 2, 1. が実行されると

−37.5␣

と印字する．印字の後 space を一つとる．

例 A〕C：DFC/519203−.

221/**PRNTFIX**, A, 3, 2.

このとき ␣␣−9.20␣

と印字する．整数部の印字桁数指定 i を実際の桁数より多く指定すると多い桁数だけspaceを先頭にとる．

8.2.4 PRNTINTEGER

使用法 221/**PRNTINTEGER**, I.

Iは整数型定数領域の foot digit を指し，その領域に格納されている整数を印字する subroutine である．

例 I〕C：DCW/1453+.

このとき 221/**PRNTINTEGER**, I.

が実行されると

␣␣1453␣␣

と印字する．印字の際整数が正のときは最初にspaceを一つとり，負のときはーを印字する．印字の後spaceを一つとる．

例 J〕C：DCW/00123ー．

このとき 221/**PRNTINTEGER**, J.

が実行されると

␣␣ー123␣

と印字する．上位桁の0は印字されずspaceとして印字される．

8.2.5 PRNTX

使用法 221/**PRNTX**, I, i.

I は整数定数領域のfoot digitを指す．

i は整数領域の下位の桁をi桁印字することを指示する．

例 I〕C：DCW/4137ー．

このとき 221/**PRNTX**, I, 3.

が実行されると

137

と印字する．この場合は符号を無視し，印字の後spaceをとらない．

例 K〕C：DCW/0032+.

このとき 221/**PRNTX**, K, 3.

が実行されると

032

と印字する．

8.2.6 PRNTSTRING

使用法 221/**PRNTSTRING**, LS.

LS は letter string 領域の foot digit を指す．letter string をそのままの形で印字する subroutine である．

例 LS〕C：DLS/A+B−C=D.

このとき 221/**PRNTSTRING**, LS.

が実行されると

A+B−C=D

と印字する．

8.2.7 CRLF

使用法 221/**CRLF**, i.

この subroutine はi回復帰改行をタイプライターに行なわせるものである．

221/**CRLF**, 3.

と書くと3回タイプライターは復帰改行を行なう．

8.2.8 SPACE

使用法 221/**SPACE**, i.

この subroutine はi個spaceをとるものである．

221/**SPACE**, 5.

と書くとspaceを5個とる．

8.3 標準函数

8.3.1 SQRT

使用法 221/**SQRT**, A, B.

A, B はそれぞれの浮動小数点定数領域の foot digit を指す．

このsubroutineはA fieldの浮動小数点定数の平方根を求め B field に格納するものである．演算後も A field の内容は変化しない．A field の内容は (<A>) ≧0 でなければならない．

例 A〕C：DFC/514. (=4.0)

B〕C：DFC/0.

のとき 221/SQRT, A, B.

が実行されると

B〕C：DFC/512. (=2.0)

となる．すなわち $\sqrt{(<A>)} \longrightarrow (<B>)$

もし(<A>)<0のときは **'SQRT impossible'** と印字されerrorとなり演算を中止する．

8.3.2 EXP

使用法 221/**EXP**, A, B.

A, B はそれぞれの浮動小数点定数領域の foot digit (符号桁) を指す．

$\mathbf{exp}(<A>) \longrightarrow (<B>)$

A field の内容は演算後も変化しない．

8.3.3 SIN

使用法 221/**SIN**, A, B.

A, B はそれぞれの浮動小数点定数領域の foot digit (符号桁) を指す．

$\mathbf{sin}(<A>) \longrightarrow (<B>)$

A field の内容は演算後も変化しない．

8.3.4 COS

使用法 221/**COS**, A, B.

A, B はそれぞれの浮動小数点定数領域の foot digit を指す．

$\mathbf{cos}(<A>) \longrightarrow (<B>)$

A field の内容は演算後も変化しない．

8.3.5 LOG

使用法 221/**LOG**, A, B.

A, B はそれぞれの浮動小数点定数領域の foot digit を指す．

$\mathrm{ln}(<A>) \longrightarrow (<B>)$

A field の内容は演算後も変化しない．

8.3.6 ANGTG

使用法 221/**ANGTG**, A, B.

A, Bはそれぞれの浮動小数点定数領域の foot digit を指す.

$\tan^{-1}(<A>) \longrightarrow (<B>)$

A field の内容は演算後も変化しない.

8.3.7 ABS

使用法 221/**ABS**, A, B.

A, Bはそれぞれの定数領域の foot digit を指す. A field とB field の桁数は必ず一致していなければならない.

例 A〕C : DFC/51357−.
B〕C : DFC/0.
このとき 221/**ABS**, A, B.
が実行されると
B〕C : DFC/51357. (=3.57)
となる.
すなわち $|-3.57| \longrightarrow 3.57$

A field の内容は演算後も変化しない.

例 I〕C : DCW/203−.
J〕C : DCW/0000.
このとき 221/**ABS**, I, J.
が実行されると
J〕C : DCW/2030.
となる.

8.4 換算函数

8.4.1 FIX

使用法 221/**FIX**, A, I.

Aは浮動小数点定数領域のfoot digitを指す.

Iは整数型定数領域のfoot digit を指す.

この subroutine は実数を整数に変換するものである. その場合実数の小数第1位を四捨五入して整数にするが, 得られた整数の桁数がI field の桁数より大きいときは **'error 9'** を印字する.

例 FL=5.
A〕C : DFC/521254+. (=12.54)
I〕C : DCW/0000.
このとき221/**FIX**, A, I.
が実行されると
I〕C : DCW/013+.
となる.

例 FL=5.
DATA〕C : DFC/55123+. (=12300.0)
J〕C : DCW/0000.
このとき 221/**FIX**, DATA, J.
が実行されると **'error 9'** となる. なぜならDATA field の実数を整数に変換すると12300となりJ field の桁数より大になるからである.

例 FL=5.
D〕C : DFC/50212+. (=0.212)
E〕C : DCW/0000.
このとき 221/**FIX**, D, E.
が実行されると
E〕C : DCW/0000.
となる.

例 FL=5.
A〕C : DFC/506382−. (=−0.6382)
L〕C : DCW/000.
このとき 221/**FIX**, A, L.
が実行されると
L〕C : DCW/01−.
となる.

8.4.2 FLOAT

使用法 221/**FLOAT**, I, A.

I は整数型定数領域の foot digit を指す.

Aは浮動小数点定数領域の foot digit を指す.

この subroutine は整数を実数に変換するものである.

例 FL=5.
I〕C : DCW/123+.
A〕C : DFC/0.
このとき 221/**FLOAT**, I, A.
が実行されると
A〕C : DFC/5312300+.
となる.

9. ERROR MESSAGE

VALSAS IVは以下に示す**8種**の error mode に属する誤りのある source program に対して, そのassemble 続行中にこれらの誤りを検出し, error message を印字する.

error 1, error 3, error 5の一部と **error 6**の場合以外はerror 検出後もassemble を続行する. その場合errorの起つた個所からperiodまでの間にある情報はassemble されず, また同一line中に2個以上の誤りがあるときは最初の誤りだけを検出する. error message としては, error mode, error の起つたline number, そして symbol に関係する場合はerror の原因となつたsymbolからなる. 演算途中のerror messageとしては**error 9**, その他がある.

9.1 Assemble 中の Error

9.1.1 error 1 ···MISS ENTER

◦source program の最初に ENTER. OPDEF. 以外の symbol がある場合

例 ENTRY.

↓

error 1 0001 entry

9.1.2 error 2 ···DUOBLE DEFINITION

◦location symbol を重複して設定した場合.

◦C : EQU/<L_0>, <L_1>, ···<L_n>. で<L_i>のいずれかが既定義 symbol であるとき.

9.1.3 error 3 ··· TOO MANY LOCATION SYMBOL

◦location symbol を250個を越えて使用した場合.

9.1.4 error 4 ···ADDRESS ERROR

◦C : EQU/<L_0>, <L_1>, ···<L_n>. で<L_0>が未定義の場合.

◦C : CAN/<L_1>, <L_2>, ···<L_n>. で<L_i>のいずれかが未定義である場合.

◦C : END/<L_0>. で<L_0>が未定義の場合.

◦Flip Flop関係命令においてB addressを正しく作成していない場合.

◦source programをEXIT. まで読み込み後, 未定義の symbolic address が残つている場合.
このときは **error 4** を印字後, 未定義 symbol と, そのために正しくsetされないでいる命令の address部のobject program 中に格納されるべき番地をすべて印字し, それが終ると無条件に Location Table の印字にうつる.

9.1.5 error 5···OP CODE ERROR

◦定義されていない mnemonic operation code を使用した場合.

◦mnemonic operation code の定義に際して同一のmnemonic codeに異なるmachine codeを定義した場合. この場合は assemble を中止する.

9.1.6 error 6···MEMORY OVERFLOW

◦object program の占有領域が $(4200)_{16}$番地を越える場合.

9.1.7 error 7···RELATIVE STEP ERROR

◦未定義symbolに対する relative step の絶対値が125より大きい場合.

◦assemble のある時点で未定義 symbolic address の総数が750個を越えた場合.

9.1.8 error 8···CONTROL ORDER ERROR

◦control order に誤りがある場合.

9.2 演算途中の Error

9.2.1 error 9···OVERFLOW

◦浮動小数点演算の途中 overflow を起した時.

◦**FIX** の subroutine を用いるとき実数が整数に変換され, それを格納する桁数が小さすぎたとき.

◦**EXP** の subroutine を用いたとき, exp(x)のxが1000より大となつた場合.

9.2.2 SQRT impossible

◦**SQRT** subroutine を用いるとき, 平方根を求めたい数が負である場合.

9.2.3 LOG impossible

◦LOG subroutine を用いるとき, 真数が負である場合.

10. PROGRAM 例

例1. data A, B を読み込んでその四則演算を行なう. S=A+B, D=A−B, P=A×B, Q=A/B

```
ENTER.
START) 221/READREAL, A.          read A
       221/READREAL, B.          read B
       340/A, S.                   A→S
       FLA/B, S.                 A+B→S
       340/A, D.                   A→D
       FLS/B, D.                 A−B→D
       340/A, P.                   A→P
       FLM/B, P.                 A×B→P
       340/A, Q.                   A→Q
       FLD/B, Q.                 A/B→Q
       221/CRLF, 1.              1回復帰改行
       221/PRNTREAL, S, 5.       print S
       221/PRNTREAL, D, 5.       print D
       221/PRNTREAL, P, 5.       print P
       221/PRNTREAL, Q, 5.       print Q
HALT) 62.
   A〕C : DFC/0.
   B〕C : DFC/0.
   S〕C : DFC/0.
   D〕C : DFC/0.
   P〕C : DFC/0.
   Q〕C : DFC/0.
C : END/START.
EXIT.
```

例2. 2次方程式の係数a, b, c を読み込み根を求める. $ax^2+bx+c=0$

```
OPDEF.
```

```
MOVE=340  MJP=271  JUMP=220  CL=633.
ENTER.
START) 221/READREAL, a.          read a
       221/READREAL, b.          read b
       221/READREAL, c.          read c
       MOVE/C2, A.               2→A
       FLM/a, A.                 2×a→A
       CL/13, B.                 clear B
       FLS/b, B.                 −b→B
       MOVE/b, D.                b→D
       FLM/b, D.                 b²→D
       MOVE/C4, S.               4→S
       FLM/a, S.                 4×a→S
       FLM/c, S.                 4ac→S
       FLS/S, D.                 b²−4ac→D
       MJP/MINUS, D.             if D<0 then go to
                                 MINUS
       221/SQRT, D, D.           √D→D
       MOVE/D, X1.               D→X1
       FLA/B, X1.                B+X1→X1
       FLD/A, X1.                (−b+√D)/2a→X1
       MOVE/B, X2.               −b→X2
       FLS/D, X2.                −b−√D→X2
       FLD/A, X2.                (−b−√D)/2a→X2
       221/CRLF, 1.              1回復帰改行
       221/PRNT, X1.             print X1
       221/CRLF, 1.              1回復帰改行
       221/PRNT, X2.             print X2
       JUMP/HALT.
MINUS) 221/ABS, D, D.            |D|→D
       221/SQRT, D, D.           √D→D
       FLD/A, B.                 −b/2a→B
       FLD/A, D.                 √D/2a→D
       221/CRLF, 1.              1回復帰改行
       221/PRNT, B.              print B
       221/PRNTSTRING, LS.       print ±i
       221/PRNT, D.              print D
HALT) 62.
      JUMP/START.
   a〕C : DFC/0.
   b〕C : DFC/0.
   c〕C : DFC/0.
  C2〕C : DFC/512.              C2=2.0
  C4〕C : DFC/514.              C4=4.0
  LS〕C : DLS/+−i.
   A〕C : DFC/0.
   B〕C : DFC/0.
   D〕C : DFC/0.
   S〕C : DFC/0.
  X1〕C : DFC/0.
  X2〕C : DFC/0
C : END/START.
EXIT.
```

例3. $A=\sum_{x=1}^{20}x^2$

```
ENTER.
START) 633/13, A.                clear A
       340/C1, x.                1→x
LOOP) 130/C20, x.
       260/TYPE, LG.             if x>20 then go to
                                 TYPE
       340/x, acc.
       FLM/x, acc.               x²→acc
       FLA/acc, A.
       FLA/C1, x.                x+1→x
       220/LOOP.                 go to LOOP
TYPE) 221/CRLF, 1.               1回復帰改行
       221/PRNTSTRING, LS.       print A=
       221/PRNT, A.              print A
       62.
  C1〕C : DFC/511.               C1=1.0
 C20〕C : DFC/5220.              C20=20.0
   A〕C : DFC/0.
   x〕C : DFC/0.
 acc〕C : DFC/0.
  LS〕C : DLS/A=.
C : END/START.
EXIT.
```

例4. $B=\sum_{y=3}^{15}\cos(0.1\cdot y)$

```
ENTER.
START) 633/13, B.                clear B
       340/C3, y.                3→y
  LOOP) 130/C15, y.
       260/TYPE, LG.             if y>15 then go to
                                 TYPE
       340/C01, acc.             0.1→acc
       FLM/y, acc.               0.1×y→acc
       221/COS, acc, acc.        cos(0.1·y)→acc
       FLA/acc, B.
       FLA/C1, y.                y+1→y
```

```
        220/LOOP.                    go to LOOP
TYPE)   221/CRLF, 1.                 1回復帰改行
        221/PRNTSTRING, LS.          print B=
        221/PRNT, B.                 print B
HALT)   62.
  C1〕 C : DFC/511.                  C1=1.0
  C3〕 C : DFC/513.                  C3=3.0
 C15〕 C : DFC/5215.                 C15=15.0
 C01〕 C : DFC/501.                  C01= 0.1
   y〕 C : DFC/0.
   B〕 C : DFC/0.
 acc〕 C : DFC/0.
  LS〕 C : DLS/B=.
C : END/START.
EXIT.
```

11. 操 作 法

11.1 VALSAS IVの操作法

1.) **VALSAS IV** のsystem tapeを磁気テープ装置に set し，その機番を〔3〕にする．

2.) **'563 0000 4200'** の命令をコンソールより行なうと **VALSAS IV** の processor が磁気テープから計算機本体の記憶装置へ $(4200)_{16}$ 番地より読み込まれる．

3.) source program tape を紙テープ読取装置に set する．つぎにコンソールより **'220 4200'** の命令を行なうとassembleを開始する．

4.) program tape の最初に **'OPDEF.'** がある場合，すなわち mnemonic operation code の定義がある場合は定義の処理が終ると **'opdef ok restart'** と印字し，一時停止するので，つぎに「**PROCEED**」のボタンを押すと assemble を開始する．

5.) C : END/<L_0>. の命令があると assemble を一時停止する．（この時 program tape のかけ換えの必要のある場合は行なう）
つぎに「**PROCEED**」のボタンを押せば再び assemble を開始する．

6.) program の最後の **'EXIT.'**（または<**comment**>）を読み込むとタイプライターは **'exit'**（または<**comment**>）と印字し assemble は完了する．

7.) 「**PROCEED**」を押すと磁気テープ装置より計算機本体へ各種の subroutine が読み込まれて停止する．

8.) 「**PROCEED**」を押すと C : END/<L_0>. の L_0番地より object program の実行に移る．

以上の順序で source programは object program に変換され，演算を開始する．

11.2 Location Table 打ち出しの操作法

VALSAS IV の processorは location symbol の type out routine を内蔵しているので，source program中のlocation symbolが object program 中でどんな actual address に対応させられているか知りたい場合，コンソールより **'220 4210'** の命令を与えると source program で使われているすべての location symbol と，そのactual address を type out する．

11.3 Memory Dump Routineの 操作法

program の debug（手直し）はしばしば起るが，その際object programを dump（吐き出す）させる必要がよくある．**VALSAS IV** ではその **Dump routine** を内蔵し，任意の番地以降の object program を印字させることができる．

その操作法はコンソールより **'220 4220 ※※※※'** の命令を与えると ※※※※ 番地以降の object program を印字する．印字様式はつぎのようになる．

番地	**object program**
0100	34010001020
010t	41010201032
0116	22001xt
・・・	・・・

上例からもわかるように任意の番地以降の object program を印字してくれるが，その様式は Word Markのついた桁の番地と，つぎにWord Markのついた桁が現われるまでの内容を一行に印字し，改行する．この Dump routine は印字を始めるとコンソールの **step key** を上げない限りいつまでも印字を続ける．だから希望の番地まで印字するのを見守り，印字したら **step key** を上げて停止させればよい．

12. む す び

VALSAS IV は One Pass 方式（load and go type）をとつたためassembleに要する時間は非常に短くなつたが，一方 system program の占める領域がかなりあり，object programの使用領域が制約をうけることは否めない．しかしいろいろな program をテストしてきたが，まだ memory overのerror は一度も起きていない．**VALSAS IV**はまだ改良したい点がいろいろある．Utility Subroutine をさらに付加することもその一つであろう．しかし一応**VALSAS**

IV は Assembler 言語として充分なものを備えていると自負している．文法体系の根本的な変更はもはや考えていない．

我々は今後この VALSAS IV が user に充分活用され，役立つことを希望すると共にご批判，ご要望を心から歓迎する．そしてさらにより良いものにしてゆきたいと思つている．

参考文献

1）富士通信機製造株式会社：FACOM—231，命令手引書
2）大阪大学工学部，城研究室：FACOM—231のための One Pass 方式による Assembler についての作成報告，(昭39)
3）山口大学工学部電子計算機室：VALSAS Ⅲ概説書，(昭40)

(昭和42年4月15日受理)