2. MUSE-system について

菅　忠義・関本彰次・魚田勝臣（三菱電機KK研究所）

1. まえがき

MUSEは三菱電機研究所で試作調整中のMELCOM—LD1のためにつくられたProgramming Programである。

その目的とするところはComputerを使うことを主目的とするUserのためにProgramするために必要とするMachineに対する知識を最小にして，極めて容易にProgramできるようにしたものである。もちろん単に使用することを目的とするといつても，一般に大きく分けて事務処理を目的とする場合と数値計算を目的とする場合とがある。MUSEはMathematical Useの略であり，後者の要求をみたすものである。

現在同様の目的で製作されたものでもつともpopularなのはI.B.M.のFORTRANであろう。UNIVACのMATH—MATICもまた同様のものである。この外に世界共通用語としてALGOLが提唱されている。当初われわれの研究室ではALGOLを採用する予定であつたが，1960年の春頃全三菱でI.B.M—7090を導入することが決定され1961年末にinstallされることにたつた。そのため709—FORTRANの講習も行われ三菱における FORTRAN 人口は急速に増加しつつある。このような外部事情の変化によりMUSEのLANGUAGEとしては709—FORTRANを採用することに決定した。もちろん7090とMELCOMとではMachineが質的にも量的にも甚だしく異つているので，LANGUAGEとしても全く同一とすることはできない。しかし機能としては殆んど同様で Capacity として制限を加えるという方針をとつた。それは709—FORTRANによるプログラムのチエツクがMELCOMで行えることを意図したためである。

MUSE は 7090 の Subcomputer として MELCOM を使用することが可能であるのみならず，単独で有効に使用しうるために 709-FORTRAN に含まれない Statements を含んでいる。

7090-FORTRAN は未だ完成されておらずわれわれが参照したのは 709-FORTRAN LANGUAGE である。

2. MUSE LANGUAGE

MUSE LANGUAGE は次のようなものに分類される。

(1) Arithmetic Formula
(2) Control Statement
(3) Subprogram Statement
(4) Input Output Statement
(5) Specification Statement

以下順次これらの概要を示す。

(1) Arithmetic Formula

Arithmetic Formula で必要となるいくつかの概念を説明する。

Constants
- o fixed point constants (integer に限る)
- o floating point constants:

例： 27.，−.001， 4.0E−7

Variables
- o fixed point variables: 1～5 characters の alphabetic 及び numerical character であり (special character ではない) その最初の文字が I,J,K,L,M 及び N である。
- o floating point variables: 1～5 character の alphabetic 及び numerical character でありその最初の文字が I,J,K,L,M 及び N でない。

o注　意

variable には Function の最後のFを除いたものと一致する name を与えてはならない。また variable name が4字より少くないなら最後にFをつけてはならない（Function name は4字以上であるから）

Subscript

o subscript は fixed point の量であつてその値は array の member を示す。Vを任意の fixed point variable としてCを符号のついていない fixed point constant とすると subscript は以下の形の何れかでなければならぬ。

V, C, V±C, C✻V, C✻V±C

o注　意

subscript として subscripted variable を用いてはならない。

Subscripted Variables

o fixed or floating point variable で name の後に括弧をもち，その中にコンマで区切つた1，2，あるいは3コの subscript がある。

o例：A (3)，ALPHA (5✻J−5, L)

Arrangement of Arrays in Storage:

系列が2次元，3次元の場合前にある subscript の方が優先されるように全体に線型順序を与えて，この順で store される。

Rules for Constructing Expressions:

arithmetic expression において floating point mode と fixed point mode とを混用してはならない。しかしこれは floating point の式に fixed point のものがあつてはならないという意味ではない。

ある一つの mode が他の mode の式の中にある特定の様式でのみ表われることが可能である。

a) floating point の数は fixed point の式の中において function の argument としてのみ用いうる。

b) floating point の式においては fixed point の数を function の argument として，また指数として用いうる。

Hierarchy of Operations:

expression の中で括弧によつて operation の順位が示されていないときは乗ベキ，乗除算，加減算の順となる。

＊＊（乗ベキ），＊（乗算），／（除算），＋（加算），－（減算）

Ordering Within a Hierarchy:

連続した乗算及び除算（または加減算）に括弧がついていないときは左から計算が行われる。

以上で arithmetic formula に必要である事柄の説明を終つたので arithmetic formula そのものについてのべる。

Arithmetic Formula:

general form	examples
" a = b " a は variable(subscripted variable でもよい) であり b は expression である。	A = B＊C + D A(I) = B(I) + E

MUSE statement での＝は“等しい”といういみではなく“右辺の結果を左辺のものにおきかえる”といういみである。
従つて

M＝M＋1

という書式も許される。

また ＝ の左辺の variable が fixed point であると計算結果は小数点以下は切捨てられる。

(2) Control Statements

Unconditional $\overline{GO}\ \overline{TO}$:

general form	example
“$\overline{GO}\ \overline{TO}$ n” n は statement number	$\overline{GO}\ \overline{TO}$ 3

Computed $\overline{GO}\ \overline{TO}$:

general form	example
“$\overline{GO}\ \overline{TO}$ (n_1, n_2, ····, n_m), i” n_1, ····, n_m は statement number i は non-subscripted fixed point variable	$\overline{GO}\ \overline{TO}$ (30, 40, 4), I

excution のときに i の値が１，２，····，m であることに応じて statement number n_1 , n_2 , ····, n_m へ control がうつる。

Assigned GO TO :

general form	example
"GO TO n (n_1, n_2, ···, n_m)" n は non-subscripted fixed point variafle で前に excute された ASSIGN statement に現われるものであり n_1, n_2, ···, n_m は statement number である。	GO TO K(12,13,18)

この statement により，その直前に ASSIGN statement で指定せられた n の値に等しい statement number の statement に control が transfer される。

ASSIGN:

general form	example
"ASSIGN i TO n" i は statement number であり n は non-subscripted fixed point variable である。そしてこの n は assigned GO TO statement に現われるものである。	ASSIGN 18 TO K

この statement は，すぐ次の GO TO n (n_1, ····, n_m) で statement number i をもつた statement へ control を transfer する。

IF:

general form	example
"IF (a) n_1, n_2, n_3" a は expression であり，n_1, n_2, n_3 は statement number である。	IF(A(I,J)—B)20,4,6

$a<0$ のときは n_1，$a=0$ のときは n_2，$a>0$ のときは n_3 へ control が transfer される。

SENSE LIGHT:

general form	example
" SENSE LIGHT i "	SENSE LIGHT 3

i は 0,1,2,3 である。$i=0$ のときはすべての sense light が消される。その他のときは i に指定された sense light のみが点火される。

IF (SENSE LIGHT):

general form	example
"IF(SENSE LIGHT i) n_1, n_2" n_1, n_2 は statement number i は 1, 2, 3, である。	IF(SENSE LIGHT 3) 10,12

sense light が on であるか off であるかによつて control が n_1 か n_2 かへ transfer される。もしも light が on であつたときにはこの statement の実行後 off になる。

IF (SENSE SWITCH)

general form	example
"IF(SENSE SWITCH) n_1, n_2" n_1, n_2は statement number	IF(SENSE SWITCH)20,13

sense switch が down なら n_1, up なら n_2 へ control が transfer される。

DO :

general form	example
"DO n $i = m_1, m_2$" "DO n $i = m_1, m_2, m_3$" nは statement number, iは non-subscripted fixed point variable m_3がないときは $m_3 = 1$ を意味する。	DO 20 I = 1 , 10 DO 20 I = 1 , M , 2

DOが statement の直ぐ次の statement から始つて statement number n をもつ statement まで計算し，これを $i = m_1$ から m_3 ずつ増加して繰返し，$i = m_2$ となるまでこの操作を続ける。これが完了したとき statement number n をもつ statement の次の statement に control が transfer される。

全 range を通じて，iは fixed point variable としてでもあるいは subscripted variable の subscript としてでも用いることが出来る。

1つの DO の range の中に他の DO statement があつてもよいが，このときは後者の range は前者の range 中に含まれなければならない。DO の深さは4迄である。

CONTINUE :

general form	example
CONTINUE	CONTINUE

これは DO の range の最後に用いられることが多い。

この外に PAUSE, STOP, END という statements がある。

(3) Subprogram Statement

MUSE では subprogram として function-type のものと subroutine-type のものを備えている。

function-type { (i) open functions
(ii) closed functions
(iii) arithmetic functions

(i) open functions とは比較的短い, subprogram で, MAMA-LANGUAGE で MUSE system 中に貯えられているもので source program にその function name があらわれると呼び出されて組みこまれるものである。従つて built-in function ということができる。

(ii) closed functions はやはり MUSE system 中に MAMA-LANGUAGE で貯えられているが source program 中その function name があらわれる度に組みこまれるのではなく, 頻度のもつとも高い唯一の位置について, その subprogram の optimize が行われるものである。

(iii) arithmetic functions とは, プログラマが任意に MUSE language で arithmetic formula の形式で定義できる functions であり, 機能的には closed-type となるものである。

これらの name は何れも 4～5 コの alphabetic あるいは numeri-

cal character (special character は除く) からなつており，最初の文字は必ず alphabet であり最後の文字はFでなければ ならない。

subroutine-type のものは，function-type で表現できないような subprogram のことであり naming や calling の方法が function-type のものと異つている。これについての詳しい説明は略する。

(4) Input-Output Statement (709-FORTRAN と異る)

in-out statement としては次のものがある。

```
READ   CARD
READ   TAPE
READ   FLEXO  TAPE
READ   MAG    TAPE

PUNCH  CARD
PUNCH  TAPE
PUNCH  FLEXO  TAPE
WRITE  MAG    TAPE

TYPE   IN
TYPE   OUT
```

これら statement の直後に FORMAT statement をおきその format に従つて in-out が行われる。

FORMAT としてはつぎのようなものが指定される。

1. fields 数 W
2. 小数点以下の digits 数 d
3. E, F, I
4. repetition

5. hallerith field

例

FORMAT(3HXy=F 8.3, 4HbbZ=F 6.2)

これはたとえば

Xy=−9 3.2 1 0　Z=−0.0 1

8d　2b　6d

(5) Specification Statement

DIMENSION, FREQUENCY, WIDTH, EQUIVALENCE, COMMON 等がある。

ここでは DIMENSION と WIDTH についてのみ説明する。

DIMENSION:

general form	example
"DIMENSION V, V, ..." Vはvariable name(array)であり，1，2，3コのunsigned fixed point subscriptをもつている。	DIMENSION A(15), B(4,6)

DIMENSION statement は array に対して object program の中で Storage を割りあてるのに必要な information を与えるものである。

DIMENSION statement 中の unsigned fixed point subscript はその subscript の最大値を示す。

WIDTH:

general form	example
" WIDTH *m* " *m* は fixed point constant	WIDTH 10

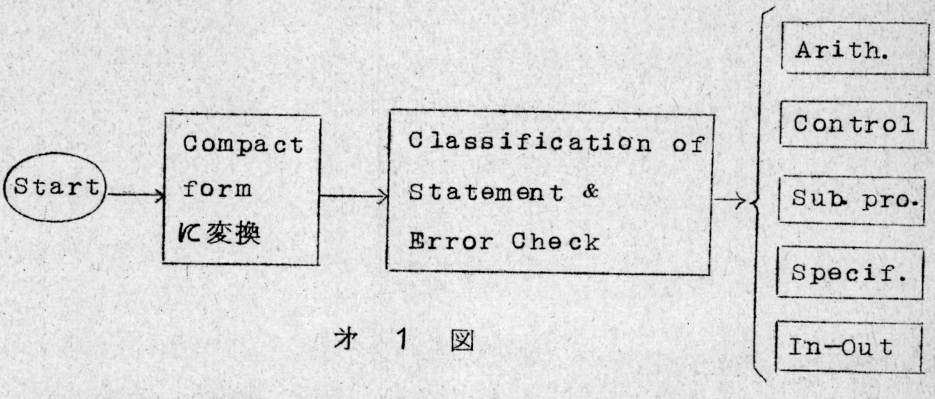
WIDTH statement は MELCOM 特有のものである。これは MELCOM の hard-wear の特徴を生かすためのものである。machine についての知識がないときはこの statement を用いる必要はなくそのときは自動的に standard width がえらばれる。

3. MUSE-System

MUSE-systen は MUSE-LANGUAGE でかかれた source program を MAMA-LANGUAGE に変換するものである。

MAMA とは Mitsubishi Automanic Minimum Access coding system の略であり，MAMA LANGUAGE は single address form の symbolic language である。MELCOM は drum を使用しており，かつ machine language として next command を指定しうる form ($2+\frac{1}{2}$ type) をもつている (参考文献4) ので object program を作製するに当つて optimize が問題になるが，MAMA system は MAMA-language から machine-language への compile を行うと同時にその optimize をも行う機能をもつている。従つて I.B.M社の SOAP に相当するものであるが SOAP とは optimize の方法が本質的に異つている。(参考文献1，2 参照)

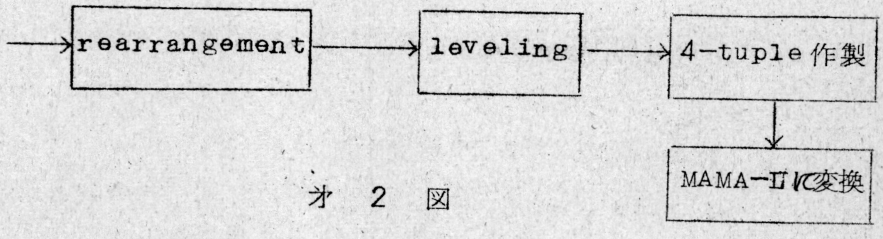
MUSE-system の flow を示すと次図の如くなる。



オ 1 図

Classification が行われてそれぞれの statement compilation routine へ行くのであるが，もつとも興味のあるのは Arithmetic formula compilation と Control statement のうち DO の compilation とであろう。

まず arithmetic formula compilation についてのべる。



オ 2 図

rearrangement では parenthese, operators, variables, constants を one word に分配する操作を行う。

つぎに各括弧に level number をつける。number のつけ方は open なら＋1，close なら－1とする。

左から search して max level number の括弧の中を左から処理して行く。4－tuple とはオ3図のような構成をもつている。

L_A	L_R	Ω	π

オ 3 図

Ωは operator，πは operand，Lは level number である。

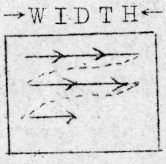
4－tuple には実際に MAMA－LANGUAGE に convert する時に必要なあらゆる情報が記録されている。問題は subscripts をもつた variables の処理である。これは DO̅ statement の処理とも関係をもつ。MELCOM においては machine が drum であり，$2+\frac{1}{2}$ type であるので DO̅ の処理にもつとも工夫を要したし 従つてまたその特徴が表われていると思われるので，これについて少しくわしくのべる。

drum memory で繰り返えし演算を行う場合 optimize がはなはだしくくずれてしまう。これを最小に止めることが重要な問題となる。そこでわれわれのとつた方法は

(i) DATAを横に並べる（WIDTH を与える）

(ii) Optimization value を求める

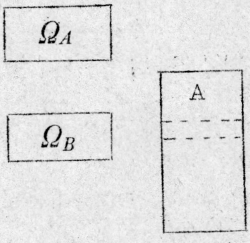
ということである。



オ 4 図

WIDTH とは DATAを横に並べるときに用いる channel の数である。（オ4図）

optimization value の説明のために いま オ5 図の場合を考えるとAというDATA がつぎつぎに pick up され図の点線の位置 以下のものを pick up 後 Ω_B を excute するには毎回1 drum 待たねばならなくなる。



オ 5 図

このような損失を防ぐためにWIDTHと array の長さから optimization value を計算しその値を Ω_A の excute time に加えたものを Ω_A の excute time とみなして，MAMA－system で optimize を行うのである。arrayの DATA を pick up するのに command modify をする必要があるが，その方法としては Index register を用いる方法と command 合成を行う方法がある。MELCOM の Index register （参考文献3 参照）は word－modify と channel modify が行えるのであるが，WIDTH という考え方を用いている関係上 Index command を有効に使用しうる場合は限られてくる。特に automatic compilation

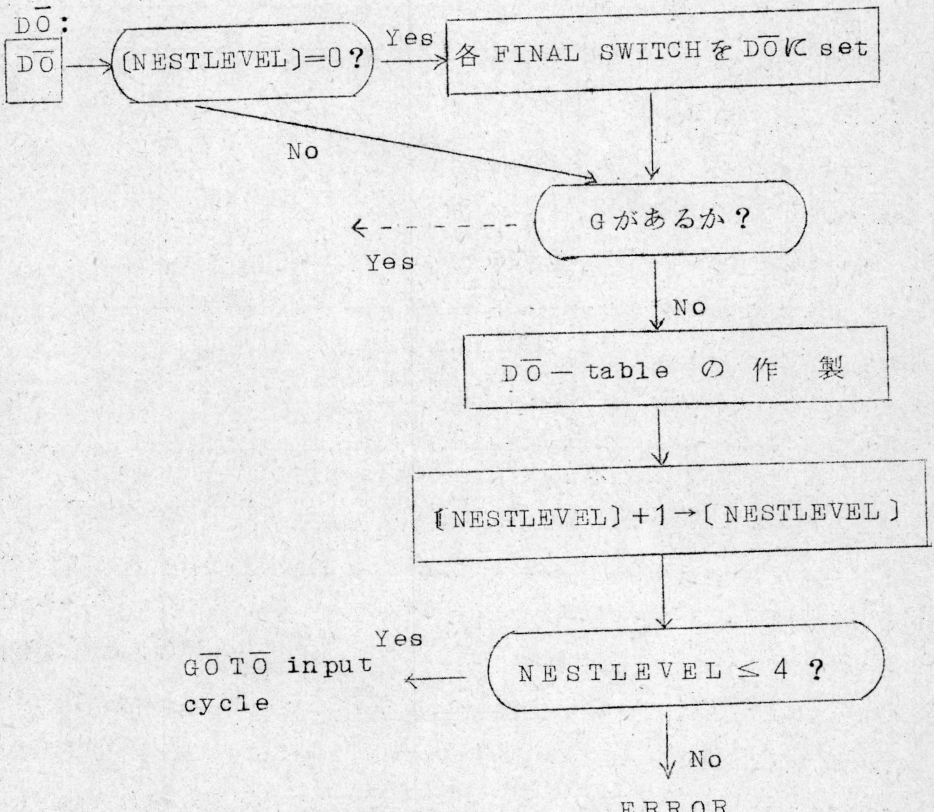
をするためには統一的方法を用いることが望ましい。そこで

(iii) 1次元の array の時は Index を用いる。

(IV) 2次元，3次元では command 合成法を用いる。

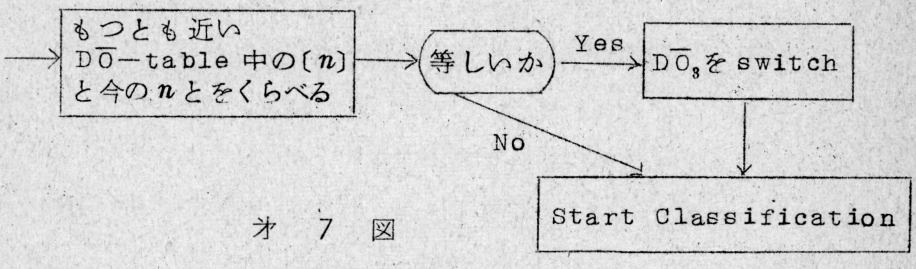
ことにした。

command 合成というのは，machine language としてARIの内容と（S.T）の内容を加えたものをARIにおいて excute するという command（JAX）があるので，（S.T）には basic command を入れておき Virtual Index register というものを設けておいて，この内容をARIへ clear and add し，ついで JAX を行つて command の modify を行う方法である。$D\bar{O}$ の flowを示すと第6, 7, 8, 9図のようにする。



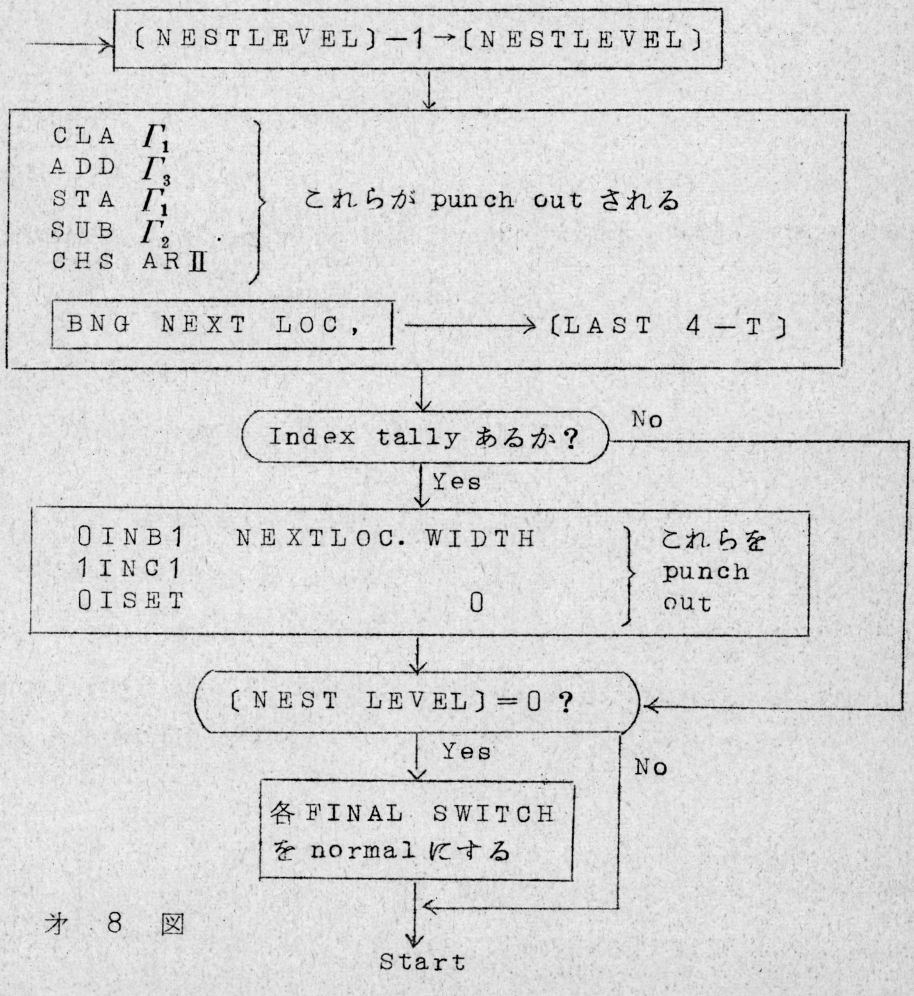
オ 6 図

$D\overline{O}_2$: (input cycle より)



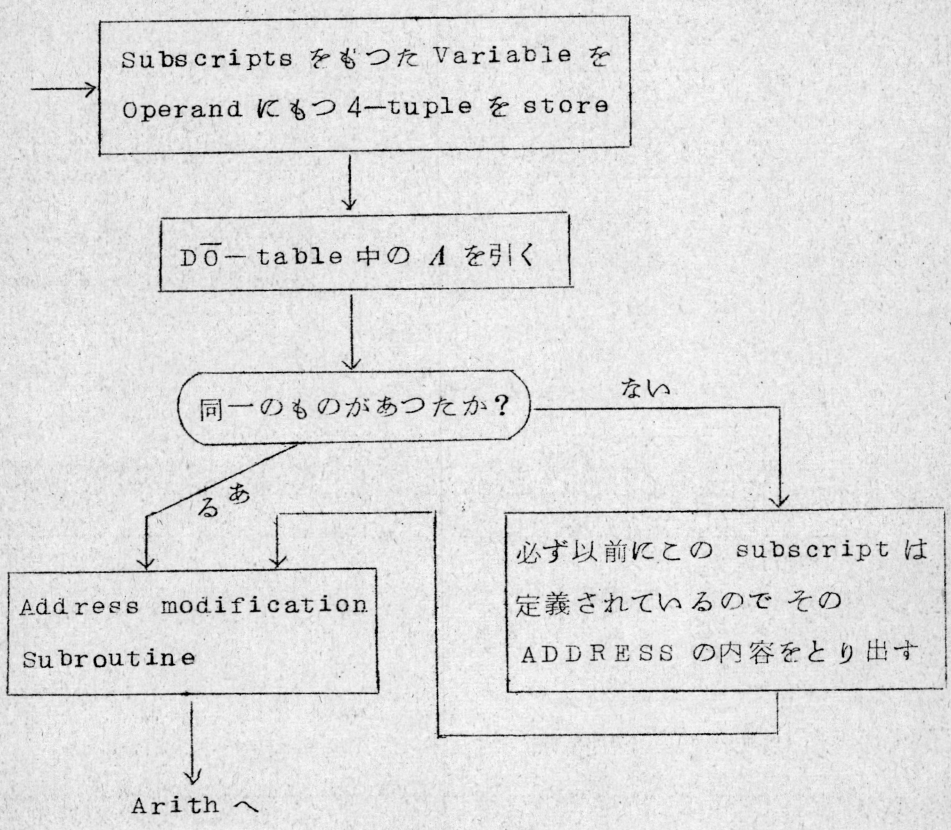
オ 7 図

$D\overline{O}_3$: (1つの DO̅-range を閉じるとき)



オ 8 図

$D\overline{O}_4$: (arith.より)



第 9 図

4. MAMA System :

MELCOM-LD 1 では MUSE と MAMA が一緒になつて object program が作製される。従つて MAMA についても説明すべきであるが, その概略は参考文献2に示してあるのでここでは省略する。

5. あとがき

MUSE は目下試作の段階であり今後も修正改良を加えて行く予定で

ある。従つてここにのべたものが最終のものではない。

MUSE の研究製作に対して絶えず激励をいただいた豊田室長に深く感謝します。

参考文献

1. 菅 “Theory of Sequator and its application for programming, 1960年, 春期数学会予稿集
2. 菅, 関本, 魚田 “MUSE及びMAMAについて” 1960年情報処理学会予稿集
3. 中塚, 前田, 壺井 “MELOM—LD1 の Index Register 方式と Out put 方式” 1960年情報処理学会予稿集
4. 豊田外 “計数形電子計算機の特殊演算高速化方式” 三菱電機 Vol.34, 1960.