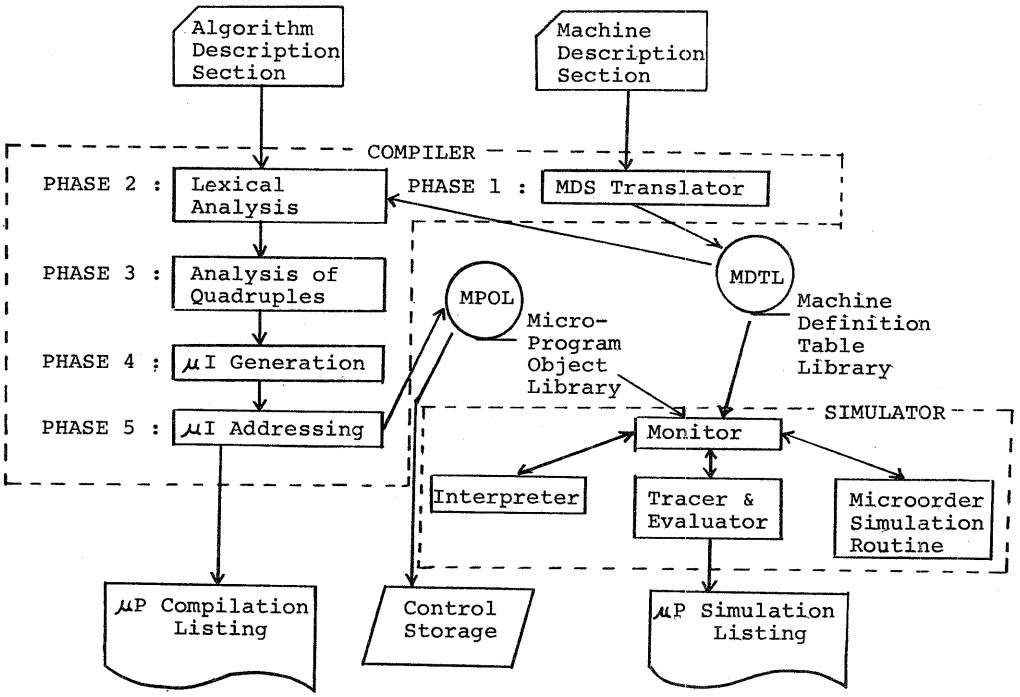
DA 27－1
（1975. 11. 19）

マイクロプログラム・ジェネレータの作成

馬場　敬信　　藤本　裕司　　萩原　宏
（電気通信大学）　（京　都　大　学）

1. はじめに

マイクロプログラム・ジェネレータ（μPG）は、マシンの記述と、マイクロプログラム（μP）のアルゴリズムの記述を入力とし、ビットパターン形式のμPおよびその論理シミュレーションの結果を出力するシステムである。図1に、このシステム構成を示す。



（図1）　マイクロプログラム・ジェネレータのシステム構成

我々は、前回[1]主としてマシン記述部の仕様とその処理について報告したが、その後、HITAC 8350を用いてシステムの作成を行ない、現在、種々の検討を加えている。そこで今回は、作成したシステムの概要と、これを使用して得られた結果について報告する。

2. マシン記述部とアルゴリズム記述部の処理

2. 1　マシン記述部の処理 ── Phase 1

マシン記述部（Machine Description Section : MDS）の処理については、前回報告したように、これを、コンパイラおよびシミュレータの使用のために、マシン定義テーブル（MDT）に変換し、磁気テープMDTLに出力する。

MDTは、表1に示したような部分テーブルと、後の検索の便宜のための各テ

ーブル間のポインタから成っている。

表1　Machine Definition Table (MDT) の構成

テーブル名	内容
FDT	μIの各フィールドの名称と、そのビット長
TIM	各フィールドに含まれるμOが実行されるタイミング
PST	パリティ・フィールドの詳細
MT	各フィールドに属するμOのビット・パターンと、それに与えるニモニック・コード
AGS	次μIへの分岐のためのアドレス生成法
VT	ハードウェア・ユニットの性質を表わし、以下6つの補助テーブルに分かれる
TCT	テスト可能なVariableと値の組
CCT	カウンタ値の増減に関する制御
FFS	フリップ・フロップのセット/リセットに関する制御
MST	メモリのRead/Writeに関する制御
SPAST	スクラッチ・パッド・メモリのアドレス生成法
TRT	Variable間の転送
CT	μIのフィールド、又はハードウェアによる定数の生成法
OPT	ALUで実行される演算

2.2 アルゴリズム記述部の処理 ── Phase 2

アルゴリズム記述部 (Algorithm Description Section : ADS) では、作成するμPのアルゴリズムを、MDSに対応した記述要素を用いて記述する。たとえば、演算や代入文に現われる変数は、レジスタやスクラッチ・パッド・メモリなどのMDSで記述したハードウェア・ユニットであり、オペレータは、MDSで定義した任意のものが使用される。この意味で、MDSはADSに対する宣言部であるということができる。ADSは、algorithm (ALG) ブロックと、procedure (PRO) ブロックから成り、各ブロック内では、演算、代入文のほか、IF文やGOTO文などを記述することができ、レジスタ・トランスファー・レベルの記述言語よりもハイレベルなものとなっている。ADSの記述例を付録図1に示す。

Phase 2では、MDTLからMDTを読み込み、次に、ADSをこれによってチェックしながら、ADSをquadruple列に変換する。quadrupleが表2に表したような形式でPhase 3以降に渡されると共に、Symbol Table (SBT) が

```
**********  SYMBOL TABLE DUMP  ***   ( FOR DEBUGGING )
  (ADDR)   SYMBOL TYPE    QN     FN     AA      STATEMENT NUMBER  ( *:DEF #:CALL )
  01D914   END    A***X   0000   0000   0A20    *0002    0018
  01D954   MULT   A***    0001   0000   0A50    *0003
  01D994   L0     L***    0004   0000   0000    *0007     0015
  01D9D4   L1     L***    0009   0000   0000     0008    *0010
  01DA14   L2     L***    0010   0000   0000     0012    *0014
**********  END OF SYMBOL TABLE DUMP  ***
```

(図2) Symbol Table

作成され．同様に Phase 3 以降で使用される。図2は．付録図1に示したADS例について作成されたSBTである。

表2　quadruple の形式と意味

（形　式）　意味	（形　式）　意味
(:=　, VAR1, VAR2,　N)	(＃　, k　, CV　,　)
VAR2の内容をVAR1に転送する。NはVAR1の個数を表わす。	カウンタCVをkだけ増減する。
(<-　, FF　, FFS　,　)	(B　,　,　,BA) (BR, VAR,　,BA)
flip-flop FFにFFSをセットする。	(BCT, C,　,BA) (BP,　,　,BA)
(BOP, OPND1, OPND2, RES)	(BCE, C, BA1, BA2) (ORP,　,　,　)
(UOP,　, OPND2, RES)	(BMC, C1, C2, BA)
OPND1, OPND2に binary (BOP) あるいは unary (UOP) operation を施し、RESに代入する。	BAで指定された quadruple への分岐を行なう。Cはテスト条件を示す。これらの quadruple を、ブランチ・タイプの quadruple と呼ぶ。

3．最適なマイクロ命令の構成

3.1 最適化のための前処理 ── Phase 3

Phase 3 では、Phase 2 の出力である quadruple 列を入力として、これをセグメントに分け、同時に、available register を決定する。ここでセグメントとは、実行がなされるときは、必ずその先頭の quadruple からシーケンシャルに最後まで行なわれる quadruple 列のことである。実際には、セグメントの先頭を、(i) ALG又はPROブロックの先頭、(ii) ラベルの付けられた quadruple、あるいは(iii) BP、BCTの次の quadruple とし、セグメントの終りをブランチタイプの quadruple としてセグメントを決定している。

available register とは、あるセグメント内で、最後に値が参照されてから、次に値を更新されるまでの、レジスタがスクラッチ・パッド・メモリのことである。

これらセグメントと available register は、次の Phase 4 で使用される。available register は、(i) 転送の途中でのデータの一時記憶、(ii) 演算結果の一時記憶などに、コンパイラが割当てる。セグメントは、Phase 4 でのマイクロオーダ（μO）の Framing に使用されると共に、シミュレータでリストへの出力を制御するパラメータとしても使用される。

付録図1に応じた quadruple 列に対して Phase 3 を実行した結果を図3に示した。

3.2 マイクロ命令の構成 ── Phase 4

Phase 4 では、quadruple 列、SBT、MDT、及び available register とセグメントを利用して、オブジェクトμPを出力する。この過程では、より少ないマイクロ命令（μI）数を目指し、後に述べる最適化を行なっている。

処理は quadruple ごとに行ない、各 quadruple に対して、(i) 指定された動作を行なう為に必要なμOをMDTから見つけ出し（μO mnemonic：MN）、(ii) その動作を行なったときに、値を参照される変数（Used Variable：UV）と、(iii) 値を更新される変数（Defined Variable：DV）を決定する。以後、quadruple に対して（MN, UV, DV）の組を作成することを generation と呼ぶ。

Phase 4 では、主要なルーチンとして、Path Search Routine（PSR）と、

```
MICRO PROGRAM GENERATOR (VER-001)     AVAILABLE REGISTER AND PROGRAM SEGMENT LISTING
DECLARED AVAILABLE REGISTER
                    AV. REG       DIMENSION
                    SPM0(39X)     '00:31'
AVAILABLE REGISTER
  SEGMENT NO.       AV. REG       DIMENSION     START QUADRUPLE NO.   END QUADRUPLE NO.
  0001                                          0001                  0003
                    G             '00:07'       0001                  0001
                    U             '00:31'       0001                  0002
                    L             '00:31'       0001                  0003
  0002                                          0004                  0006
                    SPM0(10X)     '00:31'       0004                  0005
  0003                                          0007                  0008
                    U             '00:31'       0007                  0008
  0004                                          0009                  000D
                    L             '00:31'       0009                  000A
                    U             '00:31'       000B                  000C
  0005                                          000E                  000F
                    L             '00:31'       000E                  000F
  0006                                          0010                  0011
  0007                                          0012                  0014
                    SPM0(10X)     '00:31'       0012                  0012
                    SPM0(11X)     '00:31'       0012                  0013
```

(図3)　セグメントとavailable register

Framing Routine (FR)を使用する。3.2.1でまずこれらについて述べ、3.2.2でgenerationについて具体的に述べる。

3.2.1 基本的なルーチン

(A) Path Search Routine

Variable間の転送路を見出すためのルーチンである。MDSで記述された、単一のμOによる個々の転送は、Transfer Table (TRT)と呼ぶテーブルとしてMDT内に表わされている。本ルーチンは、TRTを検索して、個々の転送を連結することにより、転送のsourceからdestinationに至る転送路と、これに必要なμO列を決定する。

このルーチンの処理アルゴリズムは、sourceをroot nodeとし、あるVariableと、それから単一のμOによって転送可能な各Variableを、それぞれ親nodeとするnodeとしたsearch treeを構成してゆくものである。sourceからdestinationへの最短経路を見出す必要から、searchはbreadth-firstによって行なう。

あるVariableから、単一のμOによって、同時に複数個のVariableへの転送が可能な場合が考えられる。従って、PSRの入力のdestinationとしては、複数個のVariableを認め、また、search treeの各nodeには、そのnodeからのdestinationを明らかにする必要がある。

図5に、PSRの流れ図を示した。ここで、OPENとCLOSEDの2つのリストを用意し、また、あるnode Nからのdestinationを D(N) で表わしている。

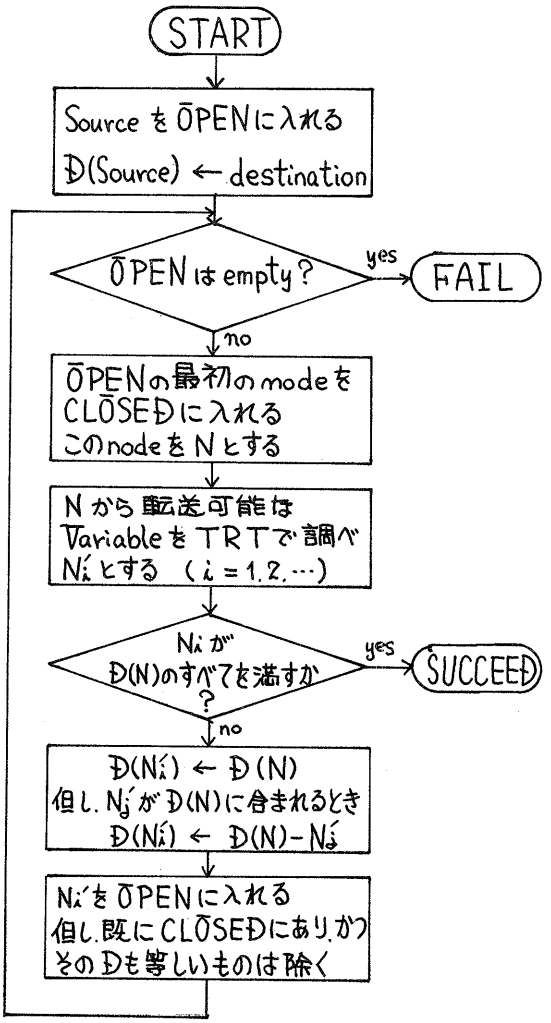
(B) Framing Routine

μPGでは、μOを組み合わせてμIを作成してゆく過程で、図4のように、FRAMEとよぶ領域を用いる。Frame Number (FN)として順に番号付けられたFRAMEの各行は、1つのμIに対応しており、Boxとよぶ μIのFieldに応じて区切られた領域と、個々の情報を格納する領域から成っている。このFRAMEにμOを書き込むことを、以下、Framingとよぶ。

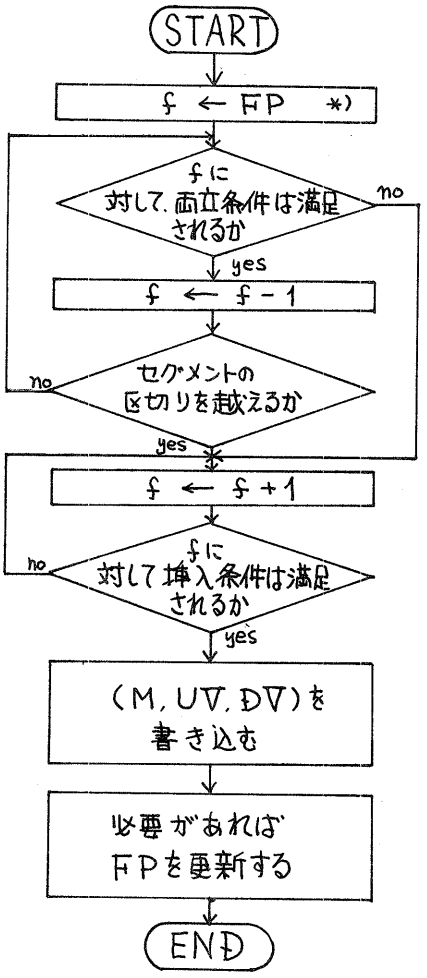
Framing Routineの入力は、generateされた(MN, UV, DV)の組であり、

FN	box1	. . . . .	box32	UV	DV	branch	. . .
1							
2							

（図４） FRAME



（図５） PSR の流れ図



*) FP：μO が既に書き込まれている Frame 行のうち FN が最大のものの FN

（図６） FR の流れ図

UV, DV には、値が参照されるタイミング, 更新されるタイミングが、それぞれ付加されている。

1つの μI に於て並列にいくつかの μO が実行される可能性を利用し、Framing Routine は、μO を組み合わせて、より少ない数の μI を構成する。この最適化は、両立条件と挿入条件とよぶ次に述べる2つの条件を判定して行なう。

(i) 両立条件

既にFRAMEに配置されているμ$\bar{0}$に対し、新たにFramingしようとするμ$\bar{0}$が、実行順序に関して矛盾を起こさないことを検証するものである。これは、既にFramingされたμ$\bar{0}$のUV, DVと、新たにFramingするμ$\bar{0}$のUV, DVに対して、そのタイミングをも考慮して判定する。

あるμIに於て、既にある(UV, DV)と、新たな(uv, dv)が両立できないとは、次のいずれかの場合をいう。

① uvと同じvariableがDVにあり、uvのタイミングがDVのタイミングより早い場合

② dvと同じvariableがUVにあり、UVのタイミングがdvのタイミングより遅い場合

③ dvと同じvariableがDVにあり、DVのタイミングがdvのタイミングより遅い場合

(ii) 挿入条件

μ$\bar{0}$が、既にFRAMEにあるμ$\bar{0}$のために、同一のμI内で同時に実行できない場合、挿入条件が満たされないという。これは、μIの同一のFieldで、排他的な2つのμ$\bar{0}$を行なわせようとした場合など、ハードウェアの制約によるものである。

Framing Routineは、Phase 3で得たセグメントの区切りを利用し、各セグメント内で既にFramingしたμ$\bar{0}$に対し、両立条件と挿入条件を満たす最も小さいFNのμIにμ$\bar{0}$をFramingする。この流れ図を図6に示す。

Framingされるμ$\bar{0}$やUV, DVは、Framing Routineを呼び出す際の再試行を可能にするために、個々にErase Contral Flagを持ち、いくつかの(MN, UV, DV)の組の選択が必要な場合、このFlagを利用することにより、Framingを行なう以前の状態にFRAMEを復帰させることができる。

3.2.2 generation

quadrupleで、ある動作が指定されたとき、この動作を行なうのに必要なμ$\bar{0}$をMDT中のテーブルから、マシンに独立なアルゴリズムで取り出す。更にgenerateされた(MN, UV, DV)をFramingすることにより、μIを構成し、可能なgenerationの中で、最少のμIに構成される(MN, UV, DV)の組を決定する。

同一のquadrupleに対して、複数個の(MN, UV, DV)の組み合わせを生じるのは次のような場合である。

(i) MDSに於て、同一のoperatorの記述が複数回行なわれた場合。これは複数個のALUが存在する場合等に起こる。

(ii) operatorが2項演算子であって、かつALUの2つの入力端子へ入力するデータを交換しても同一の結果が得られると判明した場合。

(iii) 転送を補って、テストあるいはFlip・Flopのセットを行なう際に、quadrupleに指定されたテスト条件、あるいはFlip・Flopのセット条件に適合するものがMDTに複数個存在する場合。

次に、具体例に沿ってgenerationの過程を説明する。

(a) SPM0(X"19") ⇒ U

図7にgenerateされた(MN, UV, DV)を示す。

U3は、X"19"をSPM0のアドレス・レジスタSPARにセットする。

MN		UV	DV
U3	0	*019'00:31' T2	U '00:31' TA
<SP>	0		
SPA	0		
NOP@SPA@NAB	0		
UI	0		

（図7）スクラッチ・パッド・メモリからレジスタUへの転送

<SP>は、SPM0の読み出しを行なうためのμ0である。SPA，NOP@SPA@NAB，UIは、PSRの出力で、SPM0のデータ・レジスタSPMRからALUA，BUSAを経て，レジスタUへの転送を行なう。この読み出しと転送によって、値が参照される変数は、タイミングT2でSPM0（X"19"）'0:31'であるが、これを＊019'00:31'と略記する。また、値が更新される変数は、レジスタUで、タイミングがTAである。タイミングT2は、SPM0の読み出しのタイミングをMSTから求めたものであり、タイミングTAは、レジスタUへの転送のためのμ0（UI）の属するField（BA）が実行されるタイミングである。

(b) U<+C>SPM0（X"11"）⇒ U

<+C>は、0PTから，UとSPM0（X"11"）の2つのオペランドを、ALUの、どちらの入力端子から入力しても同一の結果が得られることが分るから、図8と図9の2通りのgenerationが行なわれる。<+C>に対するALUの入力端子はALUAとALUBであり、ALUを制御するμ0は、DB，¬C1@¬C2@¬C4である。図8の場合、まずUからALUAへの転送を行なうことを試みる。この際、pathの途中でavailable registerSPM0（X"39"）を経由し、演算はSPM0（X"39"）とSPM0（X"11"）の間で行なわれることになる。U，NAB，U7，<SP>までが、available registerへの転送，U1からDB，¬C1@¬C2@¬C4までがSPM0（X"39"）とSPM0（X"11"）とを、それぞれ読み出して演算するためのμ0である。これらをFramingすると、U1とU7が同一のFieldに属するため、挿入条件を満足せずFramingが失敗となる。一方図9は，UからはALUB，SPM0（X"11"）からはALUAへの転送を試みた場合でFramingまで行なえる。結局図9が選択され、これ

MN		UV	DV
U	0	U '00:31' T5	*039'00:31' TB
NAB	0		
NOP@NAA@NAB	0		
SPI	0		
U7	0		
<SP>	0		
U1	0	*011'00:31' T2	SC '01:01' T8
<SP>	0	*039'00:31' T2	
SBB	0		
NAB	0		
U7	0		
<SP>	0		
SPA	0		
DB,¬C1@¬C2@¬C4	0		

（図8）レジスタUとスクラッチパッドメモリとの演算－1

MN		UV	DV
U1	0	*011'00:31' T2	SC '01:01' T8
<SP>	0	U '00:31' T5	
SPA	0		
U	0		
NAB	0		
DB,¬C1@¬C2@¬C4	0		

(図9) レジスタUとスクラッチ・パッド・メモリとの演算 ー2

MN		UV	DV
U1	0	*011'00:31' T2	SC '01:01' T8
<SP>	0	U '00:31' T5	U '00:31' TA
SPA	0		
U	0		
NAB	0		
DB,¬C1@¬C2@¬C4	0		
UI	0		

(図10) 選択された(MN, UV, DV)

にレジスタUへの転送を付け加えて図10となる。

(C) SC'0' = B"0"の判定

VTに付属するTestable Condition Table (TCT)を検索し、テスト条件に適合するμ0としてSC0を、UVとしてSC'0' <T3>をgenerateする。結果を、図11に示す。

(D) カウンタGのデクリメント

VTに付属するCounter Control Table (CCT)を検索し、MNとしてDG1, UV, DVとしてG'0:7' <T9>をgenerateする。結果を図12に示す。

generationの結果をframingした時のFRAMEの内容の一部を図13に示す。

MN		UV	DV
SC0	0	SC '00:00' T3	

(図11) フリップ・フロップSCのテスト

MN		UV	DV
DG1	0	G '00:07' T9	G '00:07' T9

(図12) カウンタGのデクリメント

4. μIの番地付け ── Phase 5

Phase 5では、以下の3つの処理を行なっている。

(i) Phase 4から出力されるFRAMEに対し、アドレス付けを行なう。

(ii) 分岐のために必要なμ0のgenerationとFramingを行なう。

(iii) コンパイラの最終的な出力として、FRAMEに作成したμPをビットパターンの形で磁気テープMPOLに書き込む。

μP制御方式計算機での次μIアドレスの生成法を分類すると次のようになる。

```
 MICROPROGRAM GENERATOR (VER-001)        GENERATED MICROINSTRUCTION LISTING
**************
** FN=0001 **
**************
 *** MNEM. ***
     BB   *                    AA   SPA 00 *              AB   NAB 00 *
     BA   GI  00 *              OP   NOP 00 *              SP   U2  00 *
     EX   *                     TS   *                     CL   *
     JA   *                     MM   *                     TP   *
     I    *
 *** SN = 00000001   *** BTYPE =         *** BTABP = 00000000
 *** QN =00000001
 *** UV ***
        *0   001F T2 00    &
 *** DV ***
        G    0007 TA 00    &
```

(図13)　FRAMEの内容

(i) μIのFieldのビットパターンをセットする
(ii) アドレス・レジスタの一部をインクリメントする
(iii) wired logicにより、一定の値をセットする
(iv) あるvariableの値をセットする
(v) テストの成立、不成立の結果をセットする
(vi) アドレス・スタックによる
これらの生成法は、AGSに記述される。

また、Phase 2の出力quadrupleのうち分岐に関するものは、次のようになる。(かっこ内は、quadruple中の分岐のタイプを表わすキャラクタに対応する。
(i) variableの値による間接分岐　(BR)
(ii) テストの成立、不成立による条件分岐　(BCT, BCE, BMC)
(iii) 無条件分岐　(B)
(iv) micro-subroutineへの分岐　(BP)
(v) micro-subroutineからの戻り　(ORP)

分岐のタイプとアドレス生成法の間には、一定の対応関係がある。Phase 5では、この対応関係をもとに、各FRAMEから分岐可能な番地を求め、これをもとに、FRAMEへの番地付けを行なっている。

μPGのオブジェクト・リスティングを付録図2に示す。MPOLに作成されたμPは、シミュレータへの入力、または、制御記憶へのローダの入力となる。

6. おわりに

現在、システムは一応の完成を見、我々は、種々の入力データにより、残されたバグの発見、あるいは、より使い易いシステムへの改良を試みている。システムはアセンブリ言語を用いて作成し、プログラム全体の大きさは約38万バイトで、これをオーバレイにより、最大所要メモリ14万バイトとしている。また、作成には約6人年を要した。今後、システムの評価を行なう予定である。

謝辞　システムの作成に協力戴いた磯谷幸夫、高橋訓の両氏に感謝する。また、日頃御援助戴く京都大学渡辺勝正助教授、並びに、電気通信大学有山正孝教授、武井健三教授に深謝する。

参考文献

1) 馬場敬信、萩原宏：　マイクロプログラム・ジェネレータ、情報処理学会計算機設計自動化研究会資料　73-8、1973

```
MICROPROGRAM GENERATOR (VER-001)                    SOURCE LISTING
   LINE-NO.      ST-NO.      SOURCE   STATEMENT
     00001                   *** SAMPLE ADS " MULTIPLICATION "
     00002           1          AVAIL SPM0(X"39") ;
     00003           2          EXTRN ALG END(X"A20") ;
     00004                   *** SPM0(X"10") : MULTIPLIER
     00005                   *** SPM0(X"11") : MULTIPLICAND
     00006           3        ALG MULT(X"A50") ;
     00007           4             G:=SPM0(X"18")'24:31' ;
     00008           5             U:=SPM0(X"19") ;
     00009           6             L:=SPM0(X"19") ;
     00010           7         L0: SPM0(X"10"):=<SPLA>SPM0(X"10") ;
     00011           8             IF SC'0'=B"0" THEN GOTO L1 ;
     00012           9             U:=U<+C>SPM0(X"11") ;
     00013          10         L1: L:=<SRLA>L ;
     00014          11             U:=<SRLA>U ;
     00015          12             IF SC'0'=B"0" THEN GOTO L2 ;
     00016          13             L:=L<+C>SPM0(X"30") ;
     00017          14         L2: >G ;
     00018          15             IF G¬=X"0" THEN GOTO L0 ;
     00019          16             SPM0(X"10"):=U ;
     00020          17             SPM0(X"11"):=L ;
     00021          18             GOTO END ;
     00022          19        GLA;
```

付録図1　Source Listing

```
MICROPROGRAM GENERATOR (VER-001)                    OBJECT LISTING                    75-10-29
          ** MACHINE:HITAC8350      ** ALGORITHM:MULT          ** DATE:10-29-75
ADDRESS  FRAME-NO. BB    AB    AA    BA    SP    OP    EX    CL    P1    TS    JA    TP    MM    I     P2
                   (6)   (3)   (3)   (5)   (5)   (5)   (5)   (7)   (1)   (7)   (6)   (3)   (2)   (1)   (1)
0A00     0002      00    0     4     01    13    00    00    00    0     28    01    0     0     0     0
0A01     0003      00    0     4     02    13    00    00    00    0     28    02    0     0     0     0
0A02     0004      00    0     4     14    10    11    00    00    1     28    03    0     0     0     1
0A03     0005      00    0     0     00    00    00    00    00    1     27    05    1     0     0     0
0A04     0006      08    0     4     01    11    0A    00    00    0     28    05    0     0     0     1
0A05     0007      09    0     0     14    17    00    00    00    1     28    06    0     0     0     1
0A06     0008      00    0     4     02    17    11    00    00    1     28    07    0     0     0     0
0A07     0009      08    0     0     14    17    00    00    00    0     28    08    0     0     0     0
0A08     000A      00    0     4     01    17    11    00    00    1     28    09    0     0     0     1
0A09     000B      00    0     0     00    00    00    00    00    1     27    0B    1     0     0     1
0A0A     000C      09    0     4     02    14    0A    00    00    1     28    0B    0     0     0     0
0A0B     000D      00    0     0     00    00    00    00    0F    1     28    0C    0     0     0     1
0A0C     000E      00    0     0     00    00    00    00    00    1     16    02    1     0     0     0
0A0D     000F      08    0     0     14    10    00    00    00    1     28    0E    0     0     0     0
0A0E     0010      09    0     0     14    11    00    00    00    1     28    20    0     0     0     0
0A50     0001      00    0     4     0A    12    00    00    00    0     28    00    0     0     0     1
                                  ** OBJECT LISTING END **
```

付録図2　Object Listing