日本の揺籃期のコンピュータのソフトウエア的復刻 －その2－*

大駒誠一†
慶應義塾大学理工学部管理工学科

1950年代後半、我が国では固有名詞のついた独自のコンピュータがあちこちで製作され，そこでは数多くのすぐれたプログラム作られた．それらが散逸してしまってはもったいないので，そのプログラムを収集し，保存し，かつ，実際に実行して検証してみるために，それぞれのソフトウエア・シミュレータを作製中である．第37回情報処理学会プログラミングシンポジウムでは，FUJIC，ETL-Mark4，PC-1について報告したが，これはその続きであり，ETL-Mark4AとK-1のソフトウエア・シミュレータについて報告する．

Many original computers were produced in the latter half of 1950's in Japan. To collect and preserve a lot of prominent programs of those days, we are developing the software simulators of such old computers. The simulators of FUJIC, ETL-Mark4 and PC-1 were reported at the 37th Programming symposium of IPSJ in January 1996. This is the successive report for simulators of ETL-Mark4a and K-1 computers.

1. 始めに

1996年1月第37回情報処理学会プログラミング・シンポジウムにおいて，FUJIC（富士写真フィルム，1956年3月），ETL-Mark4（電気試験所，1957年11月）およびPC-1（東京大学，1958年3月）の3個の国産のコンピュータのソフトウエア・シミュレータについて報告したが，これはその続きである．その後，ETL-Mark4A（電気試験所，1959年8月）とK-1（慶應義塾大学，1960年4月）のソフトウエア・シミュレータを作製したので報告する．

2. ETL-Mark4A

ETL-Mark4Aは，当時の通産省電気試験所（現在の電総研）において高橋茂等によって1957年11月に完成して稼動していた，トランジスタを主要素子とするETL-Mark4を改造したコンピュータである．ETL-Mark4の語長が10進5桁+符号だったものを10進7桁+符号に拡張し，メモリは磁気ドラム1000語に加えて磁気コア1000語を増設した．さらに，インデックス・レジスタをつけ，文字データをあつかえるようにした．この改造は1959年8月に完了し，その当時，PC-2（東京大学，1960年）が出現するまで国産最高速のコンピュータであった．ETL-Mark4A本体は現在上野の国立科学博物館で保管している．

*The Software Simulators for Japanese Computers in the Cradle, No.2

†Seiichi Okoma, The Administration Engineering Department of the Faculty of Science and Technology Keio University

夏のプログラミング・シンポジウム
「コンピューティングの歴史」 1996. 7. 23～25

2.1 記憶装置と演算速度

磁気ドラム　1000 語（0～999 番地）
　　呼出し時間　2ms(15,000rpm)
磁気コア　　1000 語 (1000～1999 番地)
　　呼出し時間　10 μ s
演算時間（呼び出し時間を含む平均値）

	磁気ドラム	磁気コア
加減算	4.2ms	0.24ms
乗除算	7.2ms	3.40ms
比較	2.2ms	0.24ms

2.2 語の構成

命令語　左側 2 桁で命令の種類を表し，続く 1 桁でインデックス・レジスタを指定する．アドレス部は 4 桁で，1000 番地未満は磁気ドラム，1000 番地以上は磁気コアを指した (図 1)．符号部は無視．

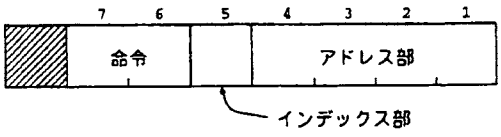

図 1　ETL-Mark4A の命令語

表 1　ETL-Mark4A の文字コード表

プリンタ				フレキソライタ		
コード	文字	コード	文字	コード	letter	figure
0	delete	40	;	0	0	P
1	space	41	.	1	1	Q
2	=	42	A	2	2	W
3	→ (¯)	43	B	3	3	E
4	*	44	C	4	4	R
5	<	45	D	5	5	T
6	.	46	E	6	6	Y
7	/	47	F	7	7	U
10	(	50	G	10	8	I
11	)	51	H	11	9	O
12	α (a)	52	I	12	π (p)	A
13	β (b)	53	J	13	%	S
14	γ (g)	54	K	14	#	D
15	δ (d)	55	L	15	-	F
16	π (p)	56	M	16	¥	G
17	θ (t)	57	N	17		bell
20	0	60	O	20		space
21	1	61	P	21		CR
22	2	62	Q	22		LF
23	3	63	R	23		letter
24	4	64	S	24		figure
25	5	65	T	25	+	H
26	6	66	U	26	-	J
27	7	67	V	27	.	K
30	8	70	W	30	,	L
31	9	71	X	31	=	Z
32	+	72	Y	32	/	X
33	-	73	Z	33	;	C
34	CRLF	74	%	34	"	V
35	≧ (>)	75		35	(	B
36	≠ (!)	76		36	)	N
37		77		37	?	M

( ) 内は，シミュレータで使用する代用文字．ASCII コードには α，π，→，≧などがないので．

```
;  ETL-Mark4Aのニュートン法による平方根のプログラム
;    作成者   大駒誠一
;    作成日   1996-05-13
;
;    サブルーチンをテストするプログラム
;    '+'が出たら、7桁の数値を入力する。
;    7桁のゼロを入力すると停止する
480a
;00;350  32t;       入力を促す'+'を出力
;01;320   7t;       7桁読み込み
;02;170  10p;       ゼロなら終わり
;03;220 100t;       a→100番地
;04;510   6p;       帰り番地をセット
;05;100 500t;       サブルーチンへジャンプ
;06;410 101t;       平方根を取り出す
;07;340   7t;       平方根を7桁出力
;08;350  34t;       改行
;09;100   0p;       戻る
;10;110   0t;       停止
;  ニュートン法による平方根サブルーチン
;  100番地の平方根を求め、101番地に入れる
500a
;00;250  10p; 帰り番地をストア
;01;410  14p; 1
;02;220 101t; 1→x
;03;270 101t; x→MDR
;04;410 100t; a
;05;080   0t; a/x
;06;530   0t; a/x→U.Acc
;07;600 101t; a/x-x
;08;540   0t; a/x-x→MDR
;09;470  15p; (a/x-x)*0.5
;10;170   0t; if zero return
;11;400 101t; (a/x+x)*0.5
;12;220 101t; (a/x+x)*0.5→x
;13;100  02p; loop
;14;9999999t; 定数   1
;15;5000000t; 定数   0.5
480g;          480番地から実行
;
;     実行例
;
;  > mark4a newton.m4a ;コマンド
;  FNi=1 [newton.m4a]
;  +0200000
;  1414213
;  +0300000
;  1732050
;  +0000001
;  0003162
;  +0000000
;
;  The time elapsed, 17 secs
```

図4　ETL-Mark4Aのプログラム例1

```
; ETL-Mark4aの1から99迄の和を求める
; プログラム
;  文献1)の23ページより
;  入力も出力もない、和は48番地に入る
040a
;0;901 99t; 99->k
;1;230 8p; 0->n
;2;410 8p; n
;3;501 0t; n+k
;4;220 8p; n=n+k
;5;995 0t; k=k-1
;6;951 2p; If k>0 jump
;7;110 0t;  stop
040g; 40番地から実行
;
;
;
; 実行終了後のダンプリスト
;   0～  39       0
;  40  9010099
;  41  2300048
;  42  4100048
;  43  5010000
;  44  2200048
;  45  9950000
;  46  9510042
;  47  1100000
;  48     4950 ; ;99迄の和
;  49～1999       0
```

図5　ETL-Mark4Aのプログラム例2

表2　ETL-Mark4A の命令一覧表 (コード順)

略号	コード	名　前	内　容
NE	00	no effect	無効
PD	02 Im	set MDR	m → MDR
RD	03	round off	10^{-8}の桁で 4 捨 5 入
EX	04	extract	[MDR]・[Acc] → Acc, MDR の各桁は 1 又は 0
DV	08	divide Acc by MDR	[Acc]/[MDR] → MQR, 余り → U.Acc
SP	09 1	absolute	|[Acc]| → Acc
SC	09 2	change sign	−1×[Acc] → Acc
J	10 In	jump	無条件ジャンプ
H	11 In	halt jump	停止
RM	12* Hm	Ch.read in	m/2 桁文字読込み, m は偶数のみ, m ≦ 98
JOV	14 In	overflow jump	オーバーフローしてたらジャンプ
JP	15 In	Acc plus jump	[U.Acc]>0 なら n にジャンプ
JM	16 In	Acc minus jump	[U.Acc]<0 なら n にジャンプ
JZ	17 In	Acc zero jump	[U.Acc]=0 なら n にジャンプ
TL	20 In	store lower Acc	[L.Acc] → n
TQM	21 In	store MQR	[MQR] → n
T	22* In	store	[U.Acc] → n
CM	24 In	clear memory	0 → n
PAM	25 In	store address	[Acc] の番地部で n の番地部を書き換える
PMA	26 In	load address	[n] の番地部を Acc の番地部に入れる
TMDR	27 In	load MDR	[n] → MDR
PMB	28 Jn	mem to ind	[n] → Ind
PBM	29 Jn	ind to mem	[ind] → n の番地部のみ書き換える
SHR	30 Im	right shift	[Acc] を m 桁右へシフト
SHL	31 Im	left shift	[Acc] を m 桁左へシフト
RN	32* Hm	No.read in	m 桁 m≦99
WN	34 I'm	output	m 桁 m≦99
WM	35 I'm	type special	文字 m を出す
RM	36 I'm	through out	m 桁 m≦99
SEL	38 Km	select I/0	K=1 なら input, K=2 なら output の機器を選択
A	40* In	add	[n]+[Acc] → Acc
AL	42* In	add to lower Acc	[n]×10^{-7}+[Acc] → Acc
ADV	44* In	add & divide	([n]+[Acc])/[MDR] → MQR
MA	46* In	mult add	[n]×[MDR]+[Acc] → Acc
AN	50* Im	raise	[Acc]+m → Acc
AQ	52*	add MQR	[Acc]+[MQR] → Acc
TD	54*	Acc to MDR	[Acc] → MDR
AD	56*	add MDR	[Acc]+[MDR] → Acc
JTB	58* Jn	Acc to ind and jump	[Acc] → Ind, n にジャンプ
S	60* In	sub	−[n]+[Acc] → Acc
SL	62* In	sub from lower Acc	−[n]×10^{-7}+[Acc] → Acc
MS	66* In	mult sub	−[n]×[MDR]+[Acc] → Acc
SDV	64* In	sub & divide	(−[n]+[Acc])/[MDR] → MQR
SN	70* Im	lower	[Acc]−m → Acc
SQ	72*	sub MQR	[Acc]−[MQR] → Acc
PA	74* Im	set address	U.Acc の番地部を m で置き換える
SD	76*	sub MDR	[Acc]−[MDR] → Acc
JPBA	78* Jn	ind to Acc and jump	[Ind] → Acc, n にジャンプ
PB	90 Jm	set ind	m → Ind
JBP	95 Jn	ind plus jump	[Ind]>0 なら n にジャンプ
JBM	96 Jn	ind minus jump	[Ind]<0 なら n にジャンプ
JBZ	97 Jn	ind zero jump	[Ind]=0 なら n にジャンプ
JRB	98 Jn	raise ind jump	[Ind]+1 → Ind, n にジャンプ
JLB	99 Jn	lower ind jump	[Ind]−1 → Ind, n にジャンプ

注) *のついている命令は、そのコードを 1 増やすとその命令実行前に U.Acc と L.Acc をクリヤーする。例えば、40 は add 命令だが、41 は clear add 命令になる。

インデックス部は，2個のインデクッス・レジスタの他，相対番地も指定できる．

数値語　メモリ上では，符号と7桁の絶対値表現で，小数点は左端にある (図2)．

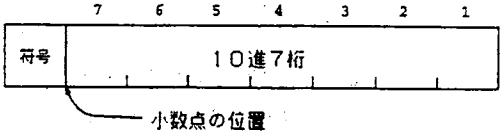

図2　ETL-Mark4A の数値語

文字語　文字の表現には10進2桁を使い，1語に3字入った．符号を含めて左側2桁は無視 (図3)．

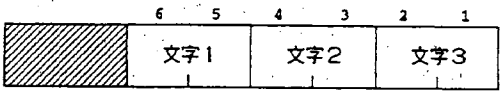

図3　ETL-Mark4A の文字語

2.3 文字コード

ETL-Mark4A では，文字コードは2種類使われた (表1)．すなわち，入力用の紙テープはフレキソライタという機械を使って作製し，印字にはこれとは別種のプリンタを使った．そして，この二つの機械のコードが違っていたために，入力と出力とでは文字コードが異なることになった．カタカナはない．

2.4 レジスタ

アキュムレータ　14桁と符号．アキュムレータ (Acc) 上では左端に小数点があるものとして演算は行なわれる．左側7桁と符号をU.Acc（アッパーアキュムレータ）といい，右側7桁をL.Acc（ローワーアキュムレータ）という．負の値は9の補数として表現する．

例えば，-0.253 は，符号も含めて

```
97469999 9999999
```

となる．

MDR　7桁と符号．乗算の乗数，除算の除数を入れる．

MQR　7桁と符号．除算の商が入る．

インデックス・レジスタ　2個あり，それぞれ4桁．符号はなく，負は9の補数で表わす．

2.5 命令の書き方

ETL-Mark4A のイニシァル・オーダーが目下未解読なので，ソフトウエア・シミュレータは仮に用意した簡単な読み込みルーチンにより命令を読み込む．今のところ，命令語の書き方は便宜上 K-1 のそれ (3.5) と同じにしてある．ETL-Mark4A のプログラムをロードする際，命令や番地部を記号で入力する習慣はなかった．むろん，プログラムを書くときには命令に略号も用いたが，それを入力用に紙テープにパンチするときには全部数字に書き換えていた．

書き方の例:

```
000 100a;   以下を100番地からロード
410 200t;   200は絶対番地
310   2t;   2はシフトの桁数
281  10p;   相対で100+10=210番地
000 100g;   100番地から実行
```

2.6　シミュレータ

ETL-Mark4A のシミュレータのプログラムはC言語で書いてあり，mark4a.c(ETL-Mark4A 固有の部分) と commonpr.c(どのシミュレータにも共通の部分) の二つのプログラム単位からなっている．これをコンパイルして実行形を作れば，昔の ETL-Mark4A のプログラムがそのまま実行できる．実行に際しては，既報告のソフトウエア・シミュレータと同様，命令トレース，番地トレース，ブレークポイント，ポストモーテムダンプなどプログラムテスト用のツールがある．図4と図5にサンプルプログラムとその実行例を示す．

3. K-1

K-1 は慶應義塾創立百年記念事業の一環として計画され，慶應義塾大学工学部内で 1958 年 7 月設計着手，同年 11 月配線組立開始，翌 1960 年 4 月稼動を始めた．Keio Centennial Computer 略して KCC とも呼ばれた．当時の通産省電気試験所のいわゆる ETL-Mark4 型のコンピュータをモデルとした 10 進数コンピュータである．トランジスタを主要素子とし，磁気ドラムを主メモリとし，1 アドレスとしても 2 アドレスとしても使用できた．トランジスタ約 1,200 個，ダイオード約 12,000 本を使用し，浮動小数点演算機構を備え，文字は英大文字の他，カタカナも使用できた．現物は現在慶應義塾大学理工学部で保管している．

3.1 記憶装置と演算速度

磁気ドラム　1200 語（0〜1199 番地）
　　呼出し時間　3ms(10,000rpm)
後に，1000 語の磁気コアメモリが追加された．
固定小数点数の演算時間（呼出し時間を除く）

加減算	0.36ms
乗除算	5.50ms

浮動小数点数の演算時間については目下不明．

3.2 語の構成

命令語　4 桁のアドレス部が 2 個あり，右側のアドレス部に通常のロード，ストアなどの対象番地，あるいは，シフトの桁数などを入れる (図 6)．左側の「次命令のアドレス部またはインクリメント部」は，4000 以上だと次命令のアドレスを指すことになり，その命令終了後そのアドレス引く 4000 番地にジャンプする．4000 未満だとインクリメントとなり，その値をアドレス部に加えた値が実効アドレスとなる．この，次の番地を指定する機構は，磁気ドラムメモリの回転待ち時間を減らすために導入されたが，これをうまく使うアセンブラがなかったために，あまり有効に利用されなかった．

```
420 4050 0500
```

は，

```
420 0000 0500; 500 番地を引算
020 0000 0050; 50 番地へジャンプ
```

の 2 命令と同じ．また，

```
420 0050 0500
```

は，

```
420 0000 0550
```

と同じで，500+50 番地が実効番地となる．インデックス部は 1 または 2 でインデックス・レジスタを指定し，4 だとアドレス部が間接番地を意味した．また，符号を負にするとブレークポイントの指定となり，コンソールのブレークポイントスイッチをオンにするとその命令のところで一時停止した．

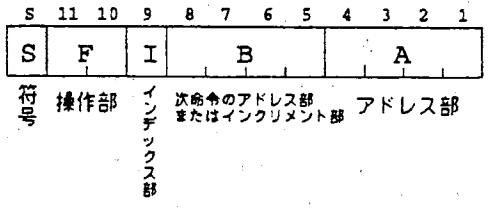

図 6　K-1 の命令語

数値語　符号と 7 桁の絶対値表現で，小数点は左端にある (図 7)．

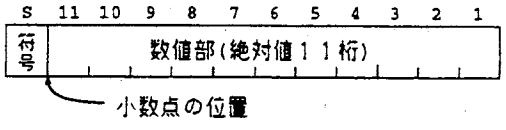

図 7　K-1 の数値語

文字語　文字の表現には 10 進 2 桁を使い，1 語に 5 字入った (図 8)．符号を含めて左側 2 桁は無視．カタカナも使えたが，カタカナの目印として符号を負とすることにし

たため，英字とカタカナは1語の中に混在できなかった(表3).

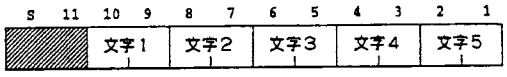

図8　K-1の文字語

3.3 レジスタ

アキュムレータ　22桁と符号．アキュムレータ(Acc)上では左端に小数点があるものとして演算は行なわれる．左側11桁と符号をU.Acc（アッパーアキュムレータ）といい，右側11桁をL.Acc（ローワーアキュムレータ）という．

Acc–Exp　2桁，符号なし．浮動小数点数の指数部を入れる．

MDR　11桁と符号．乗算の乗数，除算の除数を入れる．

MQR　11桁と符号．除算の商が入る．

MQR–Exp　2桁，符号なし．浮動小数点数の指数部計算用作業場所．

インデックス・レジスタ　2個あり，それぞれ4桁．符号はなく，負は9の補数で表現する．

3.4 浮動小数点演算

浮動小数点数はメモリ上では指数部2桁，仮数部9桁で，指数部には50下駄をはかせている(図9)．メモリに格納されるときは常に正規化され，仮数部の左端に小数点が来るよう指数部が調整される．

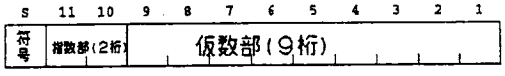

図9　K-1のメモリ上の浮動小数点数

例：　-51500000000

は，

$$-0.5 \times 10^{(51-50)} = -5.0$$

を表す．

浮動小数点数はアキュムレータ上では，メモリ上とはまったく違った表現形式を用い，指数部はAcc–Expという専用のレジスタに入れ，仮数部はアキュムレータの中央部の18桁を使用した(図10)．そのためアキュムレータに対するロード，ストア命令も浮動小数点用の加減乗除同様，整数演算とは別の命令を使用した．

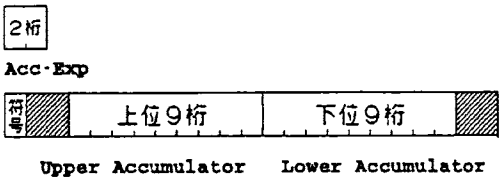

図10　アキュムレータ上の浮動小数点数

3.5　命令の書き方

命令部を記号で書くアセンブラに相当するするものはあったが，あまり使用されなかった．表4に命令の一覧表をあげる．命令語の部分はすべて数字だけで書き，必ずエンドマークをつける．

書き方の例：

```
000 100a;     以下を100番地からロード
410 200t;     200は絶対番地
280   2t;     2はシフトの桁数
481  10p;     相対で100+10=210番地
510 4026 0050p; 演算後26番地へジャンプ
000 100g;     100番地から実行
```

最初に，命令部とインデックス・レジスタの番号をつないだ3桁を書き，その後に何桁かのアドレス部が続く．途中に適当に空白を置いてよい．エンドマークが'a'なら命令のロード開始番地，'g'なら命令実行開始番地である．また，エンドマークが't'なら絶対番地，'p'なら相対番地である．図11にプログラム例，図12その実行結果を示す．

注釈　';'から';'までと，';'から改行まで

表3　K-1の文字コード表

コード	文字	コード	文字	コード	文字	コード	文字
0	AS	40	×				
1	SP	41	,				
2	=	42	A	102	ア		
3	→ (~)	43	B	103	イ		
4	*	44	C	104	ウ		
5	<	45	D	105	エ		
6	.	46	E	106	オ	146	ヒ
7	/	47	F	107	カ	147	フ
10	(	50	G	110	キ	150	ヒ
11	)	51	H	111	ク	151	ホ
12	α (a)	52	I	112	ケ	152	マ
13	β (b)	53	J	113	コ	153	ミ
14	γ (g)	54	K	114	サ	154	ム
15	δ (d)	55	L	115	シ	155	メ
16	π (p)	56	M	116	ス	156	モ
17	θ (t)	57	N	117	セ	157	ヤ
20	0	60	O	120	ソ	160	ユ
21	1	61	P	121	タ	161	ヨ
22	2	62	Q	122	チ	162	ラ
23	3	63	R	123	ツ	163	リ
24	4	64	S	124	テ	164	ル
25	5	65	T	125	ト	165	レ
26	6	66	U	126	ナ	166	ロ
27	7	67	V	127	ニ	167	ワ
30	8	70	W	130	ヌ	170	ヲ
31	9	71	X	131	ネ	171	ン
32	+	72	Y	132	ノ	172	゛
33	-	73	Z	133	ハ	173	゜
34	CR						
35	≧ (>)						
36	↑						
37	6M						

( ) 内は，シミュレータで使用する代用文字．ASCII コードには α，π，→，≧などがないので．

カタカナのコードは3桁であるが，これは紙テープの穴に対応しており，カタカナを RC(Read Character) 命令で読む込むと，U.Acc には各文字の右側2桁だけが入り，1文字でもカタカナがあると Acc の符号は負となった．したがって，カタカナと英字や数字の混在は許されなかった．

表4　K-1の命令一覧表

略号	コード	名　前	内　容
A	40*	(Add)	[U.Acc]+[n'] → U.Acc
B	42*	(Subtract)	[U.Acc]−[n'] → U.Acc
MA	44*	(Multiply Add)	[Acc]+[MDR]×[n'] → Acc
MB	46*	(Multiply Subtract)	[Acc]−[MDR]×[n'] → Acc
D	10	(Divide)	[Acc]/[MDR] → MQR, 余り → U.Acc
AD	60*	(Add Divide)	([Acc]+[n'])/[MDR] → MQR, 余り → U.Acc
BD	62*	(Subtract Divide)	([Acc]−[n'])/[MDR] → MQR, 余り → U.Acc
RO	12	(Round)	|[Acc]|+0.5×10^{-12} → U.Acc, 0 → L.Acc
R	64*	(Raise)	[U.Acc]+N' → U.Acc
L	66*	(Lower)	[U.Acc]−N' → U.Acc
SC	21	(Sign Change)	−1×[Acc] → Acc
SL	28	(Shift Left)	$10^{N'}$×[Acc] → Acc
SR	29	(Shift Right)	$10^{-N'}$×[Acc] → Acc
FA	50*	(Floating Add)	[Acc]+[n'] → Acc
FB	52*	(Floating Subtract)	[Acc]−[n'] → Acc
FMA	54*	(Floating Multiply Add)	[Acc]+[MDR]×[n'] → Acc
FMB	56*	(Floating Multiply Subtract)	[Acc]−[MDR]×[n'] → Acc
FD	11	(Floating Divide)	[Acc]/[MD] → Acc
FAD	70*	(Floating Add Divide)	([Acc]+[n'])/[MDR] → Acc
FBD	72*	(Floating Subtract Divide)	([Acc]−[n'])/[MDR] → Acc
EXR	14	(Acc-Exp Raise)	[Acc]×$10^{N'}$ → Acc
EXL	15	(Acc-Exp Lower)	[Acc]×$10^{-N'}$ → Acc
FRO	13	(Floating Round)	|[Acc]|+0.5×10^{Exp-10} → U.Acc, 0 → L.Acc
T	48*	(Store)	U.Acc → n'
FT	58*	(Floating Store)	U.Acc → n'
TA	68*	(Store Address)	U.Acc → n' の番地部
LA	78*	(Load Address)	[n'] の番地部 → U.Acc の番地部
LMD	76	(Load MDR)	[n'] → MDR
AMD	16	(Acc to MDR)	[U.Acc] → MDR
FAMD	17	(Floating Acc to MDR)	[U.Acc] → MDR
TMQ	74	(Store MQR)	MQR → n'
MQA	91	(MQR to Acc)	MQR → U.Acc
HJ	01	(Halt and Jump)	停止
J	02	(Jump)	無条件ジャンプ
JP	03	(Jump Plus)	[U.Acc]>0ならジャンプ, 0はジャンプしない
JM	04	(Jump Minus)	[U.Acc]<0ならジャンプ, 0はジャンプしない
JZ	05	(Jump Zero)	[U.Acc]=0ならジャンプ
JN	06	(Jump Non-Zero)	[U.Acc]≠0ならジャンプ
IN	80	(Index Non-Zero Jump)	[Ind]≠0ならジャンプ
IZ	81	(Index Zero Jump)	[Ind]=0ならジャンプ
IP	82	(Index Plus Jump)	[Ind]>0ならジャンプ
IN	83	(Index Non-Zero Jump)	[Ind]≠0ならジャンプ
IU	84	(Index Unequal Jump)	[Ind1]≠[ind2]ならジャンプ
IJ	85	(Set Index Jump)	SCC → Ind, n' → SCC
SI	92	(Set Index)	[N'] → Ind
LI	86	(Load Index)	[n'] の番地部 → Ind
TI	87	(Store Index)	Ind → n' の番地部
IA	90	(Index to Acc.)	Ind → U.Acc の番地部
AI	98	(Acc to Index)	U.Acc の番地部 → Ind
IR	88	(Index Raise)	[Ind]+N' → Ind
IL	89	(Index Lower)	[Ind]−N' → Ind
RW	22	(Read Word)	命令語を読む
RN	24	(Read Numeric)	数字をN'桁U.Accに読む
RC	25	(Read Character)	文字をN'桁U.Accに読む
WN	26	(Write Numeric)	U.Accの数字をN'桁出力
WC	27	(Write Character)	N'のコードの文字を出力
E	20	(Extract)	MDRの奇数の入っている桁のU.Accを消す
EE	93	(Extract Equal)	MDRとU.Accの等しい桁をU.Accに残す

注1) n'は修飾後の番地、N'は修飾後の番地部の値。
注2) *のついている命令は、そのコードを1増やすとその命令実行前にU.AccとL.Accをクリヤーする。例えば、50はFloating Add命令だが、51はClear Floating Add命令になる。

```
;        Ｋ１でのｅを計算するプログラム(ETL-Mark4のプログラムを移植)
;        インデックスレジスタを使い、1語に10桁入れる。
;        PC-9821Neで1000桁求めるのに645秒
;          作成者     大駒誠一
;          作成日     1996-05-10
00040a
;      求めたい桁数に応じて40番地(n)と41番地(m)の2語の定数を次のように変える
;      他はいっさい直す必要ない
;              桁   10   20   50  100  500 1000 2000  求めたい桁数
;40;000 253t; n    13   22   41   70  253  450  808  1/n! まで
;41;000 050t; m     1    2    5   10   50  100  200  m語
;42;000 001t; constant 1
;43;000 000t; constant 0
;  配列 e[1]からe[m]までをクリヤー
00070a
;00;862 041t;  m->Ind[2]
;01;861 043t;  0->Ind[1]
;02;491 200t;  e[i]をクリヤー
;03;881 001t;  i=i+1
;04;840 002p;  if i<m
;05;020 100t;  表をe[m]まで全部クリヤーしたら主ルーチンへ
;        主プログラム、200番地からm語にe-2(=1/2!+1/3!+1/4!+...+1/n!)を求める
000100a
;00;862 041t;  m->Ind[2]
;01;410 42t;   1
;02;480 23p;   r=1
;03;861 43t;   0->k(Ind[1])
;04;760 40t;   n->MDR
;05;411 200t;  e[k]
;06;290 10t;   e[k]/10000000000
;07;400 023p;  r+e[k]/10000000000
;08;290 001t;  r*10000000000+e[k]
;09;100 000t;  (r*10000000000+e[k])/n(nはMDRにある)
;10;480 023p;  remainder->r
;11;741 200t;  quotient->e[k]
;12;881 001t;  k+1->k(Ind[1])
;13;840 005p;  if k<m?(Ind[1]=k, Ind[2]=m)
;14;410 40t;   n
;15;660 003t;  n=n-3<=0?
;16;040 140t;  終わりなら出力ルーチンへ
;17;640 002t;  n=n+2(結果的にn=n-1)
;18;480 40t;   n=2まで
;19;030 00p
;        出力ルーチン  200番地からm語出力
000140a
;00;270 22t;   '2'
;01;270 06t;   '.'
;02;862 041t;  m->Ind[2]
;03;861 000t;  k=0
;04;411 200t;  e[k]
;05;280   1t;  11桁の内左端の1桁を消す
;06;260 10t;   e[k]を10桁出力
;07;270 001t;  1語10桁出力する毎に空白
;08;881 001t;  k+1->k(Ind[1])
;09;901 000t;  k->Accへ
;10;641 000t;  kを2倍して下1桁を調べる
;11;760 20p;   次の命令用のマスク
;12;200 00t;   kの下1桁を取り出す
;13;030 17p;   下1桁が0でなかった
;14;270 34t;   5語50桁毎に改行
;15;270 01t;   空白
;16;270 01t;   空白
;17;840 04p;  if k<m?(Ind[1]=k, Ind[2]=m)
;18;270 34t;   最後の改行
;19;010 200t;  停止
;20;000 001t;  マスク用定数
00070g
```

図11　プログラム例(eの計算)

```
2.7182818284 5904523536 0287471352 6624977572 4709369995
     9574966967 6277240766 3035354759 4571382178 5251664274
     2746639193 2003059921 8174135966 2904357290 0334295260
     5956307381 3232862794 3490763233 8298807531 9525101901
     1573834187 9307021540 8914993488 4167509244 7614606680
     8226480016 8477411853 7423454424 3710753907 7744992069
     5517027618 3860626133 1384583000 7520449338 2656029760
     6737113200 7093287091 2744374704 7230696977 2093101416
     9283681902 5515108657 4637721112 5238978442 5056953696
     7707854499 6996794686 4454905987 9316368892 3009879312
```

図12　実行結果(500桁)

は注釈としてシミュレータはこれを読み飛ばす.

3.6 プログラムの読み込み

K-1 には，命令読み込み専用とも言うべき，RW(Read Word) 命令があり，この命令を含むわずか次の 3 語だけをイニシァル・オーダーに代わるものとして常に 0〜2 番地に入れておいた．0 番地からスタートさせると，この 3 語だけで，上記の形式で書いたプログラムを所定の位置に読み込むことができた.

```
;0;220 0000 0000; 1 語分 Acc に読み込む
;1;482 0000 0000; Ind.2 の指す番地に格納
;2;882 4000 0001; Ind.2 に 1 足し 0 番地へ
```

4. 公開

これまでに作製してきた FUJIC，ETL-Mark4，PC-1，ETL-Mark4a，K-1 の 5 種類のコンピュータのソフトウエア・シミュレータの資料やプログラムなどは，いずれもインターネットの www で公開している．そのホームページアドレスは
http://www.comp.ae.keio.ac.jp/pub/fukkoku/
である．ただ，これらはいずれも今のところ未完成であり，少しずつ改善を続けている．また，上記 5 機種以外のソフトウエア・シミュレータも順次手掛けたいと考え，資料を収集中である.

5. 謝辞

ここで報告した ETL-Mark4A と K-1 のソフトウエア・シミュレータを作製するにあたって，高橋茂，夏目英雄，西野博二，渕一博，近藤頌子，原田賢一の方々に多くの資料を提供して頂いたり，こまごまとした質問に答えて頂いた．文献からだけでは得られない貴重な情報は極めて有効であった．深甚なる謝意を表する.

6. 参考文献

1) 著者不明：電子計算機とそのプログラミング，発行所不明 (1962).
2) 著者不明：ETL MARK-4A 使用の手引，第 4 版，電気試験所電子計算機研究室，(1962).
3) 高橋茂：トランジスタ計算機 (ETL-Mark III〜VI), Vol.17, No.2, pp.133-141, (1976).
4) 高橋茂:情報処理学会歴史特別委員会編日本のコンピュータの歴史，オーム社，pp.138-154 (1985).
5) 都築東吾：トランジスタ計算機の論理設計，修士論文，慶應義塾大学工学部 (1959).
6) 北川節，都築東吾:全トランジスタ形ディジタル電子計算機 K-1 について，電気通信学会雑誌，Vol.42, No.11(1959)
7) 著者不明：電子計算機 K-1 プログラム説明書，慶應義塾大学工学部中央試験所電子計算機室 (1964).
8) 著者不明：電子計算機 K-1 プログラム説明書練習問題解答集，慶應義塾大学工学部中央試験所電子計算機室 (1964).
9) 著者不明：電子計算機 K-1 命令説明書，慶應義塾大学工学部中央試験所電子計算機室 (1964).
10) 北川節：直列同期方式計算機に関する研究，学位請求論文，慶應義塾大学工学部 (1967).
11) 大駒誠一：日本の揺籃期のコンピュータのソフトウエア的復刻，第 37 回情報処理学会プログラミング・シンポジウム報告集，情報処理学会,Vol.37(1996).
12) 高橋茂：コンピュータクロニクル，オーム社 (1996).
13) 遠藤諭：計算機屋かく戦えり，アスキー (1996).