セル構造の計算機に対するプログラミング法*

菅 田 一 博** 森 田 憲 一**

Abstract

One programming method for the primitive cellular computer was given. Here, the programming means how to make an array of the instruction assemblies which were constructed with Neumann's 29-state cells by C. Lee. This presented method is very simple and systematic to make a programming. We also made it sure by PDP-12 computer simulation that this method was available.

The programming method is, in general, not limited to the cellular computer. This can be applied to the primitive computer which has the same five instructions as given by H. Wang.

1. はしがき

大規模なシステムを取り扱っていく上で「構造⇐⇒機能」の間がどのような関係になっているかは非常に興味ある問題である．この問題に対する一つのアプローチとして，Neumann によって考察された細胞オートマトンがある．これは簡単にいえば，同一の構造を持った多数の有限オートマトンを空間的に規則正しく接続したシステムのモデルである．彼の問題意識は，このような等質なオートマトンのシステムがいかに複雑な仕事を成しうるかということであった．

Neumann は彼の29状態細胞モデルでもって，チューリング機械の機能と自己増殖の機能をあわせもったシステムのモデルを作り上げた[1)2)5)6)]．これによって Neumann は細胞オートマトンの論理的万能性や自己増殖すなわち製造の万能性に対する一つの解答を与えた．

C. Lee はチューリング機械の動作をさらに基本的な論理動作に分解した動作をする原型計算機を Neumann の細胞モデルを使って構成した[3)]．この際採用した基本的な論理動作は H. Wang が考えた5種類の動作[4)]であり，C. Lee はこの5種類の機能のモジュールおよび記憶のモジュールを29状態細胞によって実現し，これらを各種の伝送ライン，ディコーダ，エンコーダと結ぶことによって原型計算機ができることを示した[3)]．

この5種類のモジュールをどのように平面上に配列すると目的とする仕事を行なうことができるか，つまり細胞オートマトンによってつくられた原型計算機のプログラミングに対して一つの組織的な方法を考えたのでそれについて報告する．また，計算機 PDP-12 によってこの原型計算機をシミュレートし，このプログラミング方法が間違ってないことを確めた．基本命令として H. Wang の5種類の機能をここで採用したのは，この5種については Neumann の29状態細胞によって非常に巧妙にモジュール化されており，Neumann の細胞さえハードウェアで経済的に作ることができれば，これで計算機をつくって働かせることができるからである．

モジュールの配列方法だけを報告しても，文献[1)2)3)]を参照しなければ実際のハードとの結びつきがわからず，細胞オートマトンによる原型計算機の全貌が把握できないと思われる．とはいえ文献[1)2)3)]はかなり膨大であり，参照するのも大変であるので必要な部分だけを取り出し説明する．したがって前半はすこし解説的であり，われわれが考えたのは第3章以後である．

2. 29状態細胞モデルによる原型計算機の構成

2.1 Lee の原型計算機

Neumann は29状態細胞モデルでもって，チューリング機械の機能と自己増殖機能を合せ持った機械を作れることを示した[1)2)]．Lee は同じ細胞を使って全

* A Programing Method for Cellular Computer, by Kazuhiro Sugata and Kenichi Morita (Faculty of Engineering Science, Osaka University)

** 大阪大学基礎工学部

く別個の考え方で原型計算機を具体的に構成した[3]. Lee はその時，機能単位として Wang が考えた5種類の論理動作を採用した[4]. それは以下のような構成になっている.

（1） square に分けられた片無限テープ.
（2） テープに書かれる文字は 0，1 の2種.
（3） テープの読み取り及び書き込み用ヘッド.
（4） プログラムが貯えられる有限メモリー.
（5） 表1に示される5種の論理動作とストップ命令.

表1　Wang による5種類の基本論理動作

命　令	記　号	動　作
Right shift	+	ヘッドを1コマ右へ移動
Left shift	−	ヘッドを1コマ左へ移動
Erase	e	ヘッドがいま読んでいるテープ上の記号を0に変える
Mark	m	ヘッドがいま読んでいるテープ上の記号を1に変える
Conditional transfer to n	t (n)	ヘッドがいま読んでいる記号が1であればn番地のインストラクションへ，0であれば次のインストラクションへ
Stop	*	ストップ

Lee はこれらを細胞モデルで実現するために，メモリー・モジュールと5種のインストラクション・モジュールを細胞で組み立てた[3]. 次にこれらを説明してゆく.

2.2 メモリー・モジュールの構成

メモリー・モジュールは図1に示されているもので，これ一つがテープの一コマに相当している. 記憶は右下部の太枠の部分に1ビット記憶される. つまり太枠の部分には図2または図3のいずれかが入り，もし図2の p(101) が入っているなら情報1が書き込まれていることを示し，図3の p(1001) が入っているなら情報0が書き込まれていることを示す. つまり，メモリー・モジュールに何が記憶されているかを外部から調べるには端子Aからパルスを一つ入れてやって端子Xからパルス列 (101) が出れば1，パルス列 (1001) が出れば0が記憶されていると判断する. これを判断するデコーダ回路は外部にある. 記憶内容を0に書き換えるときには，図2の端子Bよりパルス列(11111)を入れ，1に書き換えるときには図3の端子Bよりパルス列 (11010) を入れてやる. なお，メモリー・モジュールの中の+の記号は信号伝達線の交叉を表わす. これはパルサーとデコーダーを組み合わせることによって作れる[3].

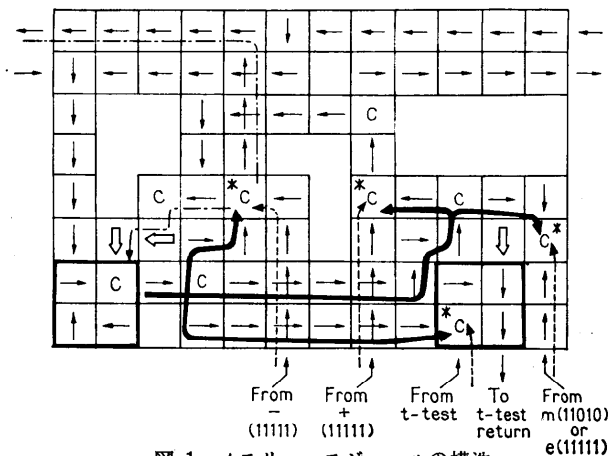

図1　メモリー・モジュールの構造

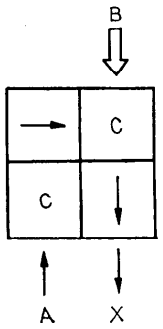

図2　represents 1
パルサー p(101)

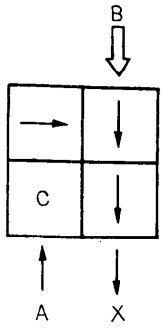

図3　represets 0
パルサー p(101)

2.3 メモリー・モジュールの動作

チューリング機械のテープに相当する機能は，この機械ではメモリー・モジュール（今後 M.M. と呼ぶ）が横につながったもの（右方に無限につながっている）が行なっている. M.M. の左下部にある circulating pulser（今後 C.P. と呼ぶ）が図4の状態をとり活性化していれば，ヘッドがその位置にあることを示す. したがって，多くの M.M. のうちで唯一つだけが活性状態の C.P. を持っている. C.P. が活性状態のとき，図1のように太線で示される道筋を通って，＊印をつけた4つのC細胞に連続したパルスが送られる. これら4つのC細胞は，この場合 AND 素子の役割をしているので，C.P. が活性化しているときのみ，----- 線で示される4つの外部からの信号を受け入れる態勢にある.

さてここで，“From−” のラインからパルス列 (11111) の信号が送られ

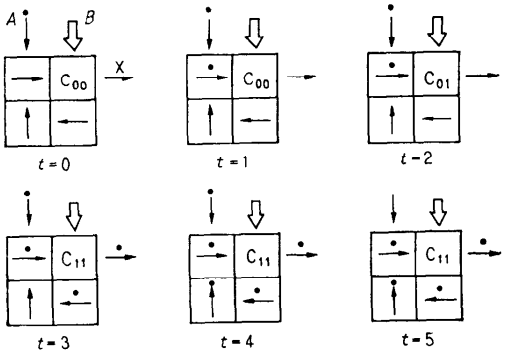

図 4 circulating pulser の活性化過程

て来た場合，図1に示されるように，一・一 線に沿って (11111) の信号が送られる．これによってまず左下部の C.P. が kill され，さらに一番上のラインを通って，一つ左の M.M. の C.P. を活性化する．つまり C.P. の活性状態の位置が1コマ左へシフトされたわけである．

“From+” のラインから (11111) が入った場合も同様に C.P. が kill され，その一つ右側の M.M. の C.P. を活性化する．

次に “t-test” のラインから（1）が入った場合は，記憶されている内容が1であるか0であるかによって “t-test return” のラインへ (101) または (1001) の信号が出てゆく．また “From m or e” のラインから (11010) または (11111) の信号が入ることによって，この M.M. の記憶内容が1または0に書き込まれる．

2.4 論理操作のモジュール

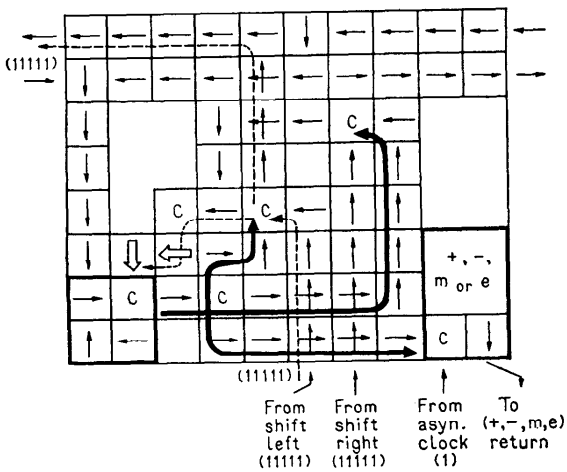

図5 一般論理操作のモジュールの構造

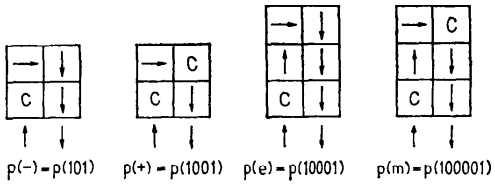

図 6 命令の種類とパルサーとの関係

原型計算機のプログラムはインストラクション・モジュールのブロックに貯えられる．インストラクション・モジュール（今後 I.M. で記述する）の種類は表1の5種の論理動作をそれぞれ行なうものがあるが，まずここでは “+”, “−”, “e”, “m” の4種の命令のモジュールを作る[3]．このモジュールは図5のようなものであり，M.M. と同様左下部に C.P. を持っている．また右下部には動作の種類に応じて図6のようなパルサーが組み込まれている．

これらのモジュールでも，C.P. が活性状態になっているときこの命令が実行され得る状態にある．まず，“From asyn clock” のラインからパルス (1) が入ると，命令の種類に応じて，(101), (1001), (10001), (100001) のパルス列が “To (+, −, e, m) return” のラインから出てゆく．このパルス列が外部の判別回路によって，何の命令であるかが判断され，その命令が実行される．次に “From shift left” のラインからパルス列 (11111) が入って来たときには，M.M. の場合と同様に，その I.M. の C.P. を kill し，その一つ左の I.M. の C.P. を活性化する．図5の ----- 線にこの過程が示される．“From shift right” のラインから入った場合も同様である．

条件付ジャンプ命令 T(n) を行なうモジュールも巧妙に作られているが，この説明は省略する．

2.5 原型計算機の動作

Lee の考えたセル構造の原型計算機の全貌を図7に示す[3]．Lee は数個所，伝送ラインを書き落したり，細部において誤った結線をしていたので訂正を加えた．図7の上部には M.M. が右方に無限に並んでおり，下部には I.M. が並んでいる．これらの中間部にそれらのモジュールと信号を授受するためのいくつかの伝送ラインがある．左側の部分には，M.M. や I.M. からの信号を解読するデコーダやパルサーがある．この計算機全体の動作は上に述べた各モジュールの動作さえわかれば，あとは逐次追っていけば理解できるので，説明を省略する．

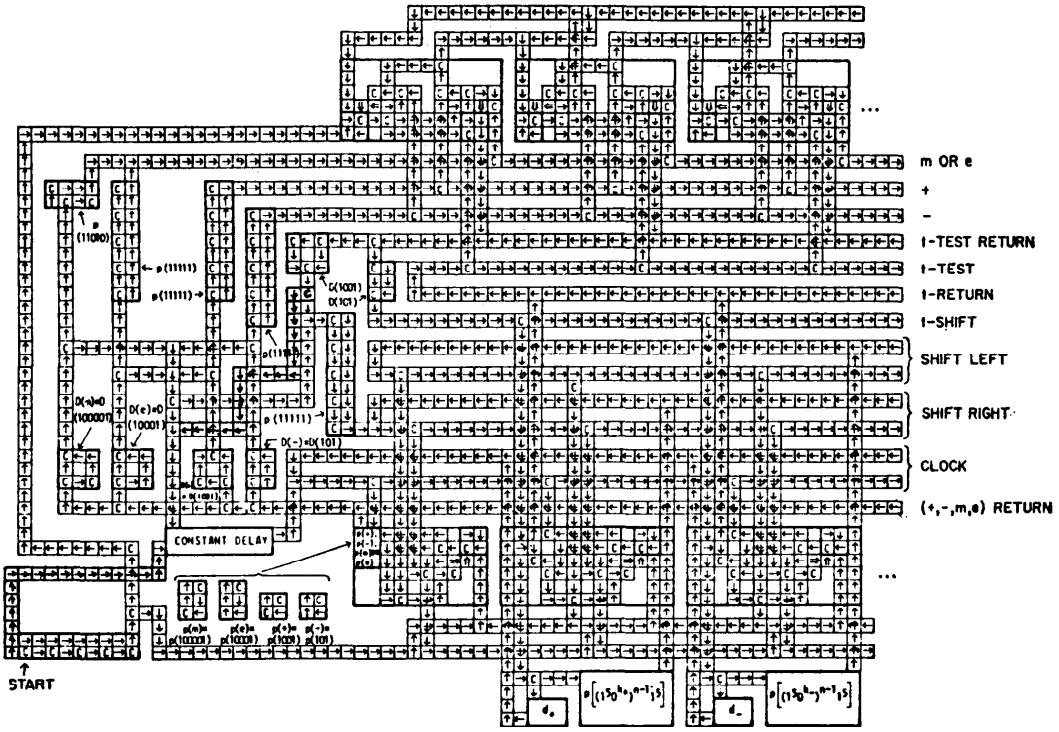

図 7 原型計算機の全体の構造

3. 原型計算機のプログラミング法

第2章で構成した原型計算機は5種類の機能を行なうモジュールを持ち，あらかじめ与えられたそれらの配列に従って計算あるいは論理的な仕事を進めていく．これらのモジュールの配列の仕方がこの原型計算機のプログラミングになる．

配列の仕方が決まったとき，実際にモジュールを並べるには5種類のモジュールを十分作っておき，これらの集りから順次適当に選んで，モジュールを右方に一列に敷きつめていく方法は一番簡単であるが面白くない．むしろ話としては，白紙状態にある細胞を敷きつめた空間に，パルス系列を適当に送りつぎつぎとモジュールの配列を生成していく方が面白い[1)2)5)]．この方法により実際に原型計算機のプログラミングが書き込めることを計算機 PDP-12 により確かめたが，これについては省略する．

ここでは細胞という概念から離れることになるが，チューリング機械が与えられたとき，原型計算機にそれと等価な働きをさせるには5種類の命令モジュールをどのように並べればよいかについて議論する[10)11)]．

5種類のモジュールを表わす記号をあらたに次のように決めておく．

E : Erase モジュール.
M : Mark モジュール.
L : Left Shift モジュール.
R : Right Shift モジュール.
T : Conditional Transfer モジュール.
(条件付飛び越し)

チューリング機械の動作は一般に5重文字 $q_iS_jS_kXq_l$ $(i, l=0, 1, \ldots, M)$ の集合で与えられる[7)8)]．どのようなチューリング機械でも2記号片無限テープのチューリング機械に変換できるので $S_j, S_k \in \{S_0=0, S_1=1\}$ とする．

(i) 与えられた5重文字表において状態が移ってゆく行き先の状態が同じ5重文字を集める．つまり5重文字 $q_iS_jS_kXq_l$ に対しては q_l に着目し，q_l に関して集めた集合を Q_l とする．

(ii) 5重文字の第一番目，第二番目の文字 q_iS_j を除外し，三番目と四番目の文字 S_kX によって，次の5つのモジュールの配列を対応させる．

0R → ERTMT

0L → ELTMT
1R → MRTMT
1L → MLTMT

この5つのモジュールの順列を，今後モジュール・ブロックと呼ぶ．

Tモジュールはジャンプする行き先によって内容的には異なっている[3]が，ここではこれらについて区別しない．以下に述べる操作（vi），（vii）で具体的に選び方が示される．

各 Q_i に含まれているそれぞれの5重文字すべてについて，この対応づけによりモジュール・ブロックをつくる．

（iii） このようにして作ったモジュール・ブロックを各 Q_i 集合ごとに一列に並べストリングをつくる．このとき各集合 Q_i のなかでのモジュール・ブロックの順序は任意でよい．このような操作で集合 Q_i から作られたモジュール・ブロックのストリングを $\vec{Q}_i$ で記述する．

（iv） 一つのストリング $\vec{Q}_i$ (l=0, 1, 2,...) のなかに，同じモジュール・ブロックが複数個存在する場合には，一つだけを残して他を除く．（これはプログラムを短くするだけのことであり，本質的ではない．）

（v） これらのストリングを $\vec{Q}_0\vec{Q}_1\vec{Q}_1\vec{Q}_2\vec{Q}_3\ldots\vec{Q}_M$ の順に一列に配列する．($\vec{Q}_0\vec{Q}_1\vec{Q}_2\vec{Q}_3\ldots\vec{Q}_M$ の順序は任意でよいがここでは説明上このように選ぶものとする．)

（vi） 操作（v）で作られたモジュールの配列中で，もし現在 $\vec{Q}_i$ の中の命令を実行しているとすると，現在の内部状態が q_i であると考える．

モジュール・ブロックは常に SXTMT (S∈{E, M} X∈{L, R}) なる配列をとっている．このうち中央にあるTは，原型計算機において C.P. が活性状態にある M.M. に情報1が保持されているときに，状態遷移する行き先にジャンプするためのものである．また一番右のTは，M.M. に情報0が保持されているときに，状態遷移の行き先にジャンプするためのものである．これは Lee のTモジュールでは M.M. の内容が0のときには，シーケンス通りに命令を実行しジャンプしないので[3][4]，TMT（仮りに $\vec{Q}_i$ の中にあるとする）の真中のモジュールMでまず M.M. に1を書き込んでこれにより遷移先の状態 q_i にジャンプする．つまり，q_i0OSXq_i を実行する．さらに，ジャンプ先のモジュール・ブロック（$\vec{Q}_i$ の中の1つ）を S′X′TMT とすれば，書き込んだ1が不要なら S′ で消されるし，必要なら再び書き込まれるので全体として不都合は起らない．

このように，この原型計算機では現在実行中のモジュール・ブロックの後方3つのモジュールTMTと，ジャンプした先のモジュール・ブロックの前2つのモジュール S′X′ とで，チューリング機械の一つの五重文字の動作を形成している．したがって，五重文字 $q_iS_jS_kXq_l$ を実行するときには，この原型計算機では状態 q_i のとき，C.P. が活性状態にある M.M. に情報 S_j が保持されていれば，まず状態 q_l に遷移し，その後に S_k を書き込み，M.M. の活性状態の位置を X 方向に移す形態をとっている．なお，この（vi）の部分は説明だけであり，プログラミングには関係ない．

（vii） 原型計算機が1を読んだとき，および0を読んだときの行き先を与えられた5重文字の動作表に従って決定し，そのような相対番地 n を持つTモジュールを選ぶ．

（viii） 初期状態が q_h であるとすれば，上で決定したモジュール配列の一番左に，Q_h にある TMT とジャンプ先が全く同じ TMT を付け加える．

（ix） チューリング機械がある状態で0を読んでストップするように与えられていると，この計算機では最後に余分な1を書き込んでしまうので，ストップする前にEモジュールを一つ入れる．

（x） これで一番左の命令モジュールより動作を始めれば与えられたチューリング機械と等価な動作をする．

4. プログラミング例

プログラミング方法を第3章に述べてきたが，説明を一般的にするために，すこし複雑になるのはまぬがれなかった．しかし，実際に例題をやってみることにより，このプログラミング方法が非常に簡潔であることを示そう．

実際にどんな論理動作をしているかを別として，例えば表2で与えられるチューリング機械を考える．プログラミングの手順は第3章で説明した順序で行なう．対応する第3章の操作番号であって，ここでのそれぞれの操作に番号をつけている．

（i） 集合 Q_0, Q_1, Q_2 は次のようになる．

$$Q_0=\{q_0\,0\,0\,R\,q_0,\qquad Q_1=\begin{cases}q_0\,1\,1\,R\,q_1\\ q_1\,1\,1\,R\,q_1\end{cases},$$

$$Q_2=\begin{cases}q_1\,0\,0\,L\,q_2\\ q_2\,1\,1\,L\,q_2\end{cases}.$$

表 2　チューリング機械の動作の一例

	0	1
q_0	$0\,Rq_0$	$1\,Rq_1$
q_1	$0\,Lq_2$	$1\,Rq_1$
q_2	＊(STOP)	$1\,Lq_2$

(ii)　それぞれの集合について，モジュール・ブロックを求めると次のようになる．

$$Q_0=\{\text{ERTMT},\qquad Q_1=\begin{cases}\text{MRTMT}\\ \text{MRTMT}\end{cases}$$

$$Q_2=\begin{cases}\text{ELTMT}\\ \text{MLTMT}\end{cases}$$

(iii)　モジュール・ブロックのストリングは次のようになる．

$\vec{Q}_0$=ERTMT.

$\vec{Q}_1$=MRTMTMRTMT.

$\vec{Q}_2$=ELTMTMLTMT.

(iv)　$\vec{Q}_1$ は次のように簡単化できる．

$\vec{Q}_1$=MRTMT.

(v)　$\vec{Q}_0$, $\vec{Q}_1$, $\vec{Q}_2$ なるモジュールの配列は次のようになる．

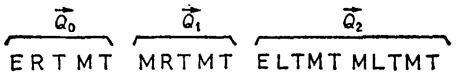

図 8　モジュールの配列

(vi)　表2によってジャンプする先を決める．ジャンプ先を矢印で表わすと次のようになる．

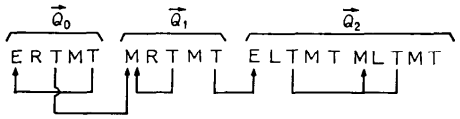

図 9　Tモジュールのジャンプ先の選択

(vii)　この場合初期状態は q_0 から始まるとし，また操作 (ix) に従ってストップする前にEモジュールを入れれば，全体のモジュールの配列は図10で示されるようになる．

これで一番左のTモジュールから動作を始めていけば表3と全く等価な計算をすることになる．

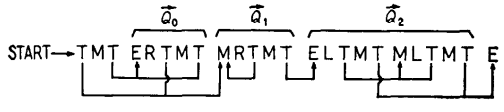

図 10　モジュール配列の全体図

5.　原型計算機のシミュレーション

ここでは，第3章に述べたプログラミング法を確めるために，2数の和を計算する原型計算機のモジュール・ストリングを生成し，計算機 PDP-12 によりシミュレーション実験をしたのでそれについて述べる．

まず，加え合わせようとする2数は次のようにタリー表現されている[9]．

整数 m: $\underbrace{11\cdots1}_{m\text{ 個}}$　　整数 n: $\underbrace{11\cdots1}_{n\text{ 個}}$

つまり，一般に整数Nを表現するには，あい続くN個のメモリー・モジュールに情報1が保持されている．このように約束すれば，加算は次のような変換を行なうことであると考えられる[7][8]．

$$\underbrace{11\cdots1}_{m\text{ 個}}0\underbrace{11\cdots1}_{n\text{ 個}}\longrightarrow\underbrace{11\cdots1}_{m\text{ 個}}0\underbrace{11\cdots1}_{n\text{ 個}}0\underbrace{11\cdots11}_{m+n\text{ 個}}$$

このような加算をするチューリング機械は表3のような動作表で与えられる[10]．このチューリング機械は加算を行うときには，最初の時点表示は次のようでなければならない．

$$0\,q_0\underbrace{11\cdots1}_{m\text{ 個}}0\underbrace{11\cdots1}_{n\text{ 個}}$$

表 3　2数の和を計算するチューリング機械の動作表

	0	1
q_0	$0\,Rq_7$	$0\,Rq_1$
q_1	$0\,Rq_2$	$1\,Rq_1$
q_2	$0\,Rq_3$	$1\,Rq_2$
q_3	$1\,Lq_4$	$1\,Rq_3$
q_4	$0\,Lq_5$	$1\,Lq_4$
q_5	$0\,Lq_6$	$1\,Lq_5$
q_6	$1\,Rq_0$	$1\,Lq_6$
q_7	＊(STOP)	$0\,Rq_8$
q_8	$0\,Rq_9$	$1\,Rq_8$
q_9	$1\,Lq_{10}$	$1\,Rq_9$
q_{10}	$0\,Lq_{11}$	$1\,Lq_{10}$
q_{11}	$1\,Rq_7$	$1\,Lq_{11}$

このチューリング機械と全く等価な動作をこれまでに述べてきた原型計算機で実行させるために，PDP-12 によりシミュレーションを行なった．その動作を PDP-12 の C.P.T. に表示し写真を撮った．その最初と最後の部分だけを取り出し，図11，図12に示した．

この写真の上部の 0，1 の数字の列が原型計算機のメモリー・モジュールに保持されている内容を表わし，また▙印が活性状態にある circulating pulser (C.P.) の位置を，その下に書かれた数字がチューリング機械

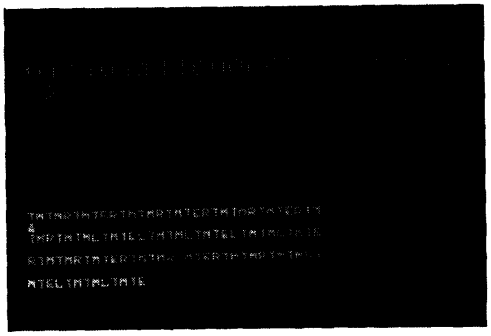

図 11 2数の和を計算する原型計算機の初期状態

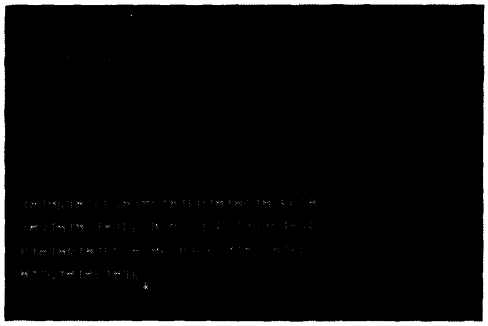

図 12 2数の和を計算し終った原型計算機の状態

との対応を示すために内部状態の番号を表現している．写真の下部のアルファベットの系列は，表3の動作表を第4章で示したプログラミング法により求めた命令系列つまり，5種類の命令モジュールの配列を示している．また，この命令系列中の小さい▙印が現在活性状態にある命令モジュールの位置を表わしている．

6. おわりに

Neumann が自己増殖するオートマトンを作るために用いた29状態の細胞モデルによって構成された，原型計算機のプログラミングの一方法を与えた．ここでのプログラミングとは29状態の細胞によって作られた機能単位（モジュール）の配列の仕方になっている．このプログラミング方法が非常に簡潔でしかも正しいことを計算機 PDP-12 によって原型計算機のシミュレーションを行なって確めた．

H. Wang の与えた5種類の命令系を持っている原型計算機[4] であるなら，特にそれが細胞モデルによって構成されていなくても，そのままこのプログラミング方法を適用できる．しかし，ここで細胞構造の原型計算機に対するものとして議論を進めたのは，C. Lee が5種類の命令を行なうモジュールを細胞モデルによって行なったものがあったからである[3]．つまり，そのモジュールをそのまま利用すれば規則的に配列された細胞モデルによって原型計算機をハードウェアとして構成でき[3]，それに対する非常に簡単なプログラミング方法が確かに存在することを示すことができたからである．

最後に本研究をするにあたり，いろいろ御示唆，御討論くださった京都大学理学部西尾英之助氏，小淵洋一氏，大阪大学基礎工学部田村博研究室の皆様に感謝します．

参 考 文 献

1) von Neumann: "Theory of Self-reproducing Automata," (A. W. Burks, ed.) Univ. of Illinois Press, Urbana, Illinois, 1966.
2) A. W. Burks: von Neumann's Self-Reproducing Automata,"Essays on Cellular Automata," (ed. A. W. Burks) University of Illinois Press (1970).
3) C.Y. Lee: "Synthesis of a Cellular Computer," Applied Automata Theory (ed. J. T. Tou), Academic Press (1970).
4) H. Wang: "A Variant to Turing's Theory of Computing Machines," J. Assoc. Comput. Mach. 4, 83 (1957).
5) 中野馨：自己増殖する機械，『数理科学』，54, [12], 1967.
6) 和田英一：フォン・ノイマンの自己増殖機械，『科学』，Vol. 37, No. 11, No. 12, 1967.
7) 渡辺茂：Turing 機械，『電気通信学会誌』，46巻11号，1963.
8) M. デービス，渡辺茂，赤摂也訳：『計算の理論』，岩波書店，1966.
9) 田中幸吉：『情報工学』，朝倉書店，1969.
10) J. W. Thatcher: Self-Describing Turing Machines and Self-Reproducing Cellular Automata, "Essays on Cellular Automata,"(ed. A. W. Burks) University of Illinois Press (1970).
11) M. A. Arbib: "Theories of abstract automata," Prentice-Hall, Inc. (1969).

(昭和46年12月14日 受付)
(昭和47年4月17日 再受付)