# 算術ステートメントの並列形直接実行装置の設計*

森下俊三** 稲垣康善*** 福村晃夫***

**Abstract**

A design and an analysis are given for the device, named PADEM, capable of parallel direct execution (execution without a compiler) of FORTRAN or ALGOL arithmetic statements. This device can execute a syntax analysis operation and arithmetic operation in parallel using two push down storages. A microprogramming technique is utilized to design it.

The processing speed and the memory capacity of this device necessary for a parallel direct execution are evaluated theoretically, and the results are checked by the simulation experiment on the computer HITAC 5020E using about 130 typical arithmetic statements.

We can expect this device, a hardware version of "one-pass-load-and-go" compiler, to decrease overhead of an operating system. The idea shown in this paper will be also applicable to a direct execution of other types of FORTRAN or ALGOL statements.

## 1. はじめに

最近の電子計算機システムの発達の陰には，システムプログラムの増大という問題があり，特に TSS を実現するためにはオーバーヘッドが大きくなってしまうのが現状のようである．そこで筆者らは，そういったシステムプログラムの一部をハードウェアにもたせてオーバーヘッドを少なくするという試みの一つとして，簡単ではあるが，この問題の本質的な部分を含むとともに，実用上も有用であると思われる算術ステートメントの直接実行装置（PADEM と呼ぶことにする）を設計した．

すでに提案されているソースプログラムの直接実行回路[1),2)]は，原理的にはコンパイラの機能をそのまま順序回路で置き換えたもので，状態数がきわめて多く，処理回路も複雑である．これに対して，本方式では，一応の処理対象は算術式に限られるが，プッシュダウン記憶 (pds) を2個使用することにより制御システムの状態数を少なくし，かつ制御機構もかなり簡単にしている．

まず最初に，**2.** では本装置 PADEM の設計を，マイクロプログラムオートマトンの立場から述べる．本装置は演算記号の並べ換え作業（変換とよぶ）と，演算実行とを同時に独立に行なうという意味で並列処理装置である．**3.** では使用する2個の pds の必要な記憶容量の評価，**4.** では従来の方式に比べて演算実行時間がどの程度短縮されるかという演算速度の評価，**5.** では科学技術計算用プログラムから選んだ算術ステートメントについての本装置のシミュレーションとその結果について述べる．

* Design of a Parallel Direct Execution Device for Arithmetic Statements, by Shunzo Morishita, Yasuyoshi Inagaki and Teruo Fukumura (Faculty of Engineering, Nagoya Uuiversity)
** 名古屋大学工学部（現在，日本電信電話公社）
*** 名古屋大学工学部電気工学教室

## 2. 並列形直接実行装置 PADEM の設計

### 2.1 設計仕様

演算記号に対して演算優先順位を定義すれば，算術ステートメントを演算実行順序に並べ直し，逆ポーランド記法による polish string を得ることができる．演算処理はこの polish string を頭から順次実行すればよい．ここではまず算術ステートメントについての仮定を述べ，次に PADEM で使用する装置とその用途および使用する命令形式について説明する．

#### 2.1.1 算術ステートメント

ここで取り扱う算術ステートメントについて，次のように仮定する．

（1） 算術ステートメントは2項演算のみからなる．

（2） 許される演算記号の集合を $\{\alpha\}$ とする．また，その演算優先順位 $\pi(\alpha)$ を **Table. 1** のように定義する．ここで，$*$, $/$, $\uparrow$ は乗算，除算，巾乗算を表わす．また，変数を一般に $v$ で表わし，その演算優先

Table 1 Operators and operational priority

| $\alpha$ | ¢, $ | = | ) | ( | +, − | *, / | ↑ |
|---|---|---|---|---|---|---|---|
| $\pi(\alpha)$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

順位を $\pi(v)=7$ と定めておく．

（3） 算術ステートメントは開始記号¢と終了記号$ とで囲まれている．

（4） 許される変数 $v$ は単純変数のみで，一文字のアルファベットまたは数字である．

### 2.1.2 使用装置と用途

PADEM で用いる装置は次のとおりである．

$P_T$: 変換用 pds

$P_E$: 実行用 pds

$R_T$ :変換用一時記憶レジスタ

$R_E$: 実行用一時記憶レジスタ

$R_B$: バッファ・レジスタ

DCD: 解読器

$N_p$: カウンタ

$F_1, F_2$: フリップ・フロップ

$P_T$ は原ストリングの演算記号の順序を演算実行順に並べ換えるのに用いられ，$P_E$ は各オペランドの記憶を行ない，かつその頭 (top) の部分で演算を実行するのに用いられる．$R_T$ は演算記号を $P_T$ へつめ込む際の，また $R_E$ はオペランドを$P_E$へつめ込む際の一時記憶レジスタである．

演算が実行されている間，$F_1$ は1状態 ($F_1=1$) をとり，この時に次に実行されるべき演算記号が検出されると制御装置は $F_2$ の状態を調べ，$F_2$ が0状態 ($F_2=0$) なら検出された演算記号を $R_B$ へ転送し，$F_2=1$ として演算終了を待つ．$F_2$ が1状態 ($F_2=1$) ならそのまま変換を停止して演算終了を待つ．その演算が終了する ($F_1=0$ になる) と，$R_B$ の内容が DCD で解読されて次の演算が実行される．$N_p$ は原ストリングを変換している各時点での括弧の深さを記憶するカウンタであり，DCD は演算記号 $\alpha$ を対応する演算開始命令 $\beta$ へ変換する解読器である．また，pds $P_T$ の頭にある演算記号の演算優先順位を $\pi(P_T)$ で，$R_T$ にある演算記号のそれを $\pi(R_T)$ で表わす．

### 2.1.3 命令形式

PADEM で用いる命令形式は次のようなものである．

（1） $f/\{X\}$: 条件 $f$ が成立する時，命令の集合 $\{X\}$ が出される．ただし $f/-$ は $\{X\}=\phi$(空集合) であり，出力命令が出ないことを意味する．

（2） $v/\rightarrow R_B$, READ: 読み込まれた記号が変数 $v$ なら，それを $R_B$ へ転送し，読み込み命令 READ を出す．

（3） $\pi(P_T)<\pi(R_T)/R_T\rightarrow P_T$, READ: 条件 $\pi(P_T)<\pi(R_T)$ が満たされると $R_T$ の内容を $P_T$ へつめ込み，入力ストリングの次の文字を読み込むための読み込み命令を出す．

（4） $R_B\rightarrow DCD(\beta)$: $R_B$ にある演算記号を解読器へ送り，対応する演算開始命令 $\beta$ を出して演算装置で実行する．

（5） $N_p\uparrow(N_p\downarrow)$: カウンタ $N_p$ の内容を1だけ増加（減少）させる．

（6） START: 入力ストリングが算術ステートメントであることがわかった時，本装置へ指示される算術ステートメント処理開始命令．

（7） READ: 入力ストリングから一文字分だけ情報を取り出し，次の一文字分の情報の取り出しの準備をする．

（8） LOCK: 一時停止命令で，状態図上の次の枝の条件 $f$ が満たされるまでその状態にとどまる．

（9） END: 処理完了命令で，全処理が完了したことを本装置から外部の装置へ知らせる．

### 2.2 状態図

本装置の状態図を **Fig. 1** に示した．状態図は大きくわけて，演算記号を $P_T$ へつめ込む部分，左括弧を $P_T$ へつめ込む部分，右括弧を検出しそれと対応する左括弧との間の部分式の実行を行なう部分，括弧で囲まれていない部分の演算記号の演算を実行する部分などからなる．また，Fig. 1 の状態図において次の約束をする．

（1） pds から情報がとり出された時，pds にあるその情報が消される際には，そのとり出し命令の右肩に＊印をつける（例 $P_T\rightarrow R_T^*$）．

（2） 各状態においてそこから出ている枝上のどの条件 $f$ も成立しない時は，エラー状態（状態 26）へはいり，かつ止る．

（3） 最終状態 25 は初期状態 0 と同じと考えてよい．

（4） 2重マルの状態は一時停止がありうる状態を示す．

（5） 各枝に書かれている数字は状態推移の所要時間である（**4.** 参照）．

〔例1〕 ¢ $a=b/c$ $

PADEM の動作を状態推移で示せば次のようにな

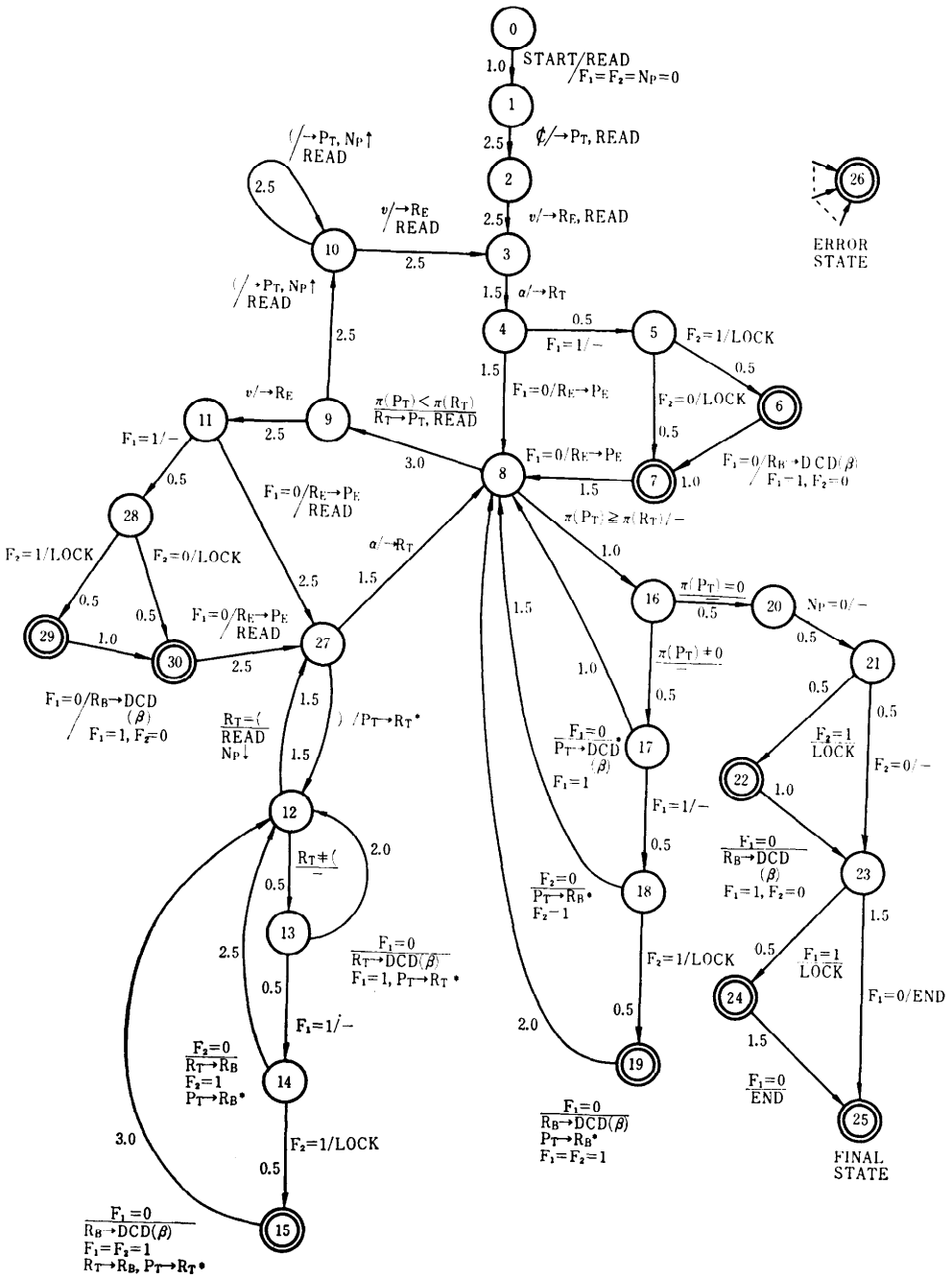
START/READ
F1=F2=NP=0
PT, READ
PT, NP↑
READ
RE
READ
RE, READ
ERROR
STATE
RT
F1=1/−
F2=1/LOCK
F2=0/LOCK
F1=0/RE→PE
π(PT)<π(RT)
RT→PT, READ
F1=0/RB→DCD
(β)
F1=1, F2=0
π(PT)≥π(RT)/−
π(PT)=0
NP=0/−
π(PT)≠0
F1=0
PT→DCD
(β)
F1=1
F2=1
LOCK
F2=0/−
F1=1/−
F1=0
RB→DCD
(β)
F1=1, F2=0
F2=0
PT→RB•
F2=1
F1=0
RT→DCD(β)
F1=1, PT→RT•
RT=(
READ
NP↓
RT≠(
F2=0
RT→RB
F2=1
PT→RB•
F1=1
LOCK
F1=0/END
F1=0
RB→DCD(β)
PT→RB•
F1=F2=1
F1=0
END
FINAL
STATE
RT→RB, PT→RT•

**Fig. 1**

る．ただし状態 $i$ から $j$ へ推移する時に，たとえば $v$ が $P_E$ へつめ込まれるならば $i(v\to P_E)\to j$ のように書く．

$0\to 1(¢\to P_T)\to 2(a\to R_E)\to 3(=\to R_T)\to 4(R_E\to P_E)\to 8(R_T\to P_T)\to 9(b\to R_E)\to 11(R_E\to P_E)\to 27(/\to R_T)\to 8(R_T\to P_T)\to 9(c\to R_E)\to 11(R_E\to P_E)\to 27(\$\to R_T)\to 8\to 16\to 17(b/c$ 実行開始$)\to 8\to 16\to 17\to 18(=\to R_B)\to 8\to 16\to 20\to 21\to 22$ ($r_1\triangleq b/c$ の実行完了まで一時停止．完了後，$a=r_1$ 実行開始$)\to 23\to 24(a=r_1$ 実行完了まで一時停止$)\to 25$(完了)．

以上のようにして，算術ステートメントは Fig. 1 の状態図で示される PADEM で直接実行できるが，ここで，DO ループ中の算術ステートメントのように，プログラム内で何度も演算実行される場合には，状態図の出力命令集合を適宜増すことにより，polish string を他の適当な場所に記憶して2度目以後の演算実行に用いれば，同じ変換を繰り返えすという欠点を避けることができることを注意しておこう．

さて，この状態図をハードウェアで実現するのにマイクロプログラミングの手法を用いることができる．状態 $i$ から状態 $j$ への推移の条件 $f$ を predicate $p_{ij}$，出力命令集合 $\{X\}$ を operator $A_j{}^i$ に対応させれば Yanov の図式が生成でき[4]，これから ALS 言語 (language of algorithmic logical schemes) による表現が得られ，マイクロプログラムオートマトン (MA) が構成できる[5]．この MA の簡約した ALS による表現 $\mathcal{A}$ は次のようになり，これは 11 個の predicates と 23 個の operators からなる*．

$$\mathcal{A}=\downarrow {}^{0}A_0\, p_0\uparrow {}^{0}A_1\, p_1\uparrow {}^{26}A_2\, p_2\uparrow {}^{26}\downarrow {}^{3}A_3\, p_3\uparrow {}^{26}A_4\, p_4\uparrow {}^{8}p_5\uparrow {}^{7}\downarrow \bar{p}_4\uparrow {}^{6}A_5\downarrow \bar{p}_4\uparrow {}^{6}\downarrow {}^{7}A_6\downarrow {}^{7}p_6\uparrow {}^{8}A_7\, p_2\uparrow {}^{81}p_7\uparrow {}^{16}\downarrow {}^{11}A_8\, p_2\uparrow {}^{3}\bar{p}_7\uparrow {}^{10}\downarrow {}^{26}A_9\omega\uparrow {}^{25}\downarrow {}^{11}A_{10}\, p_4\uparrow {}^{27}p_5\uparrow {}^{30}\downarrow {}^{29}\bar{p}_4\uparrow {}^{29}A_{11}\downarrow {}^{30}\bar{p}_4\uparrow {}^{30}\downarrow {}^{27}A_{12}\downarrow {}^{32}p_3\uparrow {}^{31}A_{13}\omega\uparrow {}^{81}\downarrow {}^{31}p_8\uparrow {}^{26}A_{14}\downarrow {}^{12}p_9\uparrow {}^{13}A_{15}\omega\uparrow {}^{32}\downarrow {}^{13}\bar{p}_4\uparrow {}^{14}A_{16}\omega\uparrow {}^{12}\downarrow {}^{14}\bar{p}_5\uparrow {}^{15}A_{17}\omega\uparrow {}^{12}\downarrow {}^{15}\bar{p}_4\uparrow {}^{15}A_{18}\omega\uparrow {}^{12}\downarrow {}^{16}p_{10}\uparrow {}^{20}\bar{p}_4\uparrow {}^{17}A_{19}\omega\uparrow {}^{81}\downarrow {}^{17}\bar{p}_5\uparrow {}^{18}A_{20}\omega\uparrow {}^{81}\downarrow {}^{18}\bar{p}_4\uparrow {}^{18}A_{21}\omega\uparrow {}^{81}\downarrow {}^{20}p_{11}\uparrow {}^{26}p_5\uparrow {}^{23}\downarrow {}^{22}\bar{p}_4\uparrow {}^{22}A_{22}\downarrow {}^{23}\bar{p}_4\uparrow {}^{23}\downarrow {}^{25}A_{23}.$$

ただし各 predicate $p_i$ は，次の条件式を意味する．$p_0\triangleq$ {start}，$p_1\triangleq\{¢\}$，$p_2\triangleq\{v\}$，$p_3\triangleq\{\alpha$(演算記号)$\}$，$p_4\triangleq\{F_1=1\}$，$p_5\triangleq\{F_2=1\}$，$p_6\triangleq\{\pi(P_T)<\pi(R_T)\}$，$p_7\triangleq\{(\}$，$p_8\triangleq\{)\}$，$p_9\triangleq\{(R_T)=(\}$，$p_{10}\triangleq\{\pi(P_T)\neq 0\}$，$p_{11}=\{N_p=0\}$，ここで $(R_T)$ は $R_T$ の内容である．また，各 operator $A_i$ は次の出力命令集合である．

$A_0\triangleq\phi$，$A_1\triangleq A_1{}^0$，$A_2\triangleq A_2{}^1$，$A_3\triangleq A_3{}^2=A_3{}^{10}$，$A_4\triangleq A_4{}^3$，$A_5\triangleq A_7{}^6$，$A_6\triangleq A_8{}^4=A_8{}^7$，$A_7\triangleq A_9{}^8$，$A_8\triangleq A_{10}{}^9$，$A_9\triangleq$ {エラー表示}，$A_{10}\triangleq A_{11}{}^9$，$A_{11}\triangleq A_{30}{}^{29}$，$A_{12}\triangleq A_{27}{}^{30}$，$A_{13}\triangleq A_8{}^{27}$，$A_{14}\triangleq A_{12}{}^{27}$，$A_{15}\triangleq A_{27}{}^{12}$，$A_{16}\triangleq A_{12}{}^{13}$，$A_{17}\triangleq A_{12}{}^{14}$，$A_{18}\triangleq A_{12}{}^{15}$，$A_{19}\triangleq A_8{}^{17}$，$A_{20}\triangleq A_8{}^{18}$，$A_{21}\triangleq A_8{}^{19}$，$A_{22}\triangleq A_{23}{}^{22}$，$A_{23}\triangleq$ {END}

ここで $A_0$, $p_0$ は表記の便宜上つけ加えたものである．

### 2.3 一般的な算術ステートメントへの拡張

前節で設けた算術ステートメントの制約をとりのぞき，一般の算術ステートメントも実行処理できるように PADEM を修正することについて論じる．

#### 2.3.1 1項演算子を含む算術式への拡張

1項演算子（ここでは unary minus のみ）が出現する部分はきわめて限定されている[3]．すなわち，unary minus (〜で示す) が現われるのは，(1)式の右辺の第1文字目か，(2)左括弧のすぐ右側かのいずれかに限る．また，binary minus や，他の unary minus と問題にしている unary minusとの相対的な位置関係により，その unary minus をキャンセルアウトするとき（例 $a-(\sim b)=a+b$）に都合がよいように，次のように優先順位を定める．

$\pi$(binary minus)$<\pi$(unary minus)$<\pi$(multiply).

また Fig. 1 に追加する状態は {31, 32} の二つ，枝は $\{d_{31}{}^2, d_{32}{}^{31}, d_3{}^{32}, d_9{}^{32}, d_9{}^{10}\}$，取り除く枝は $\{d_3{}^2\}$ である．ここで枝 $d_j{}^i$ は状態 $i$ から $j$ への枝を示す．また，修正する部分のみを **Fig. 2** に示す．

さらに，たとえば sin, cos などのような関数も，それらを一種の1項演算とみることができるが，それらを含む算術ステートメントの処理に先立って，まず最初にこのような1項演算を実行処理しておくようにすることによって，さらに直接実行できる算術ステートメントの範囲を拡げることができる．

#### 2.3.2 その他の拡張

2文字以上からなる変数や添字つき変数への拡張は，状態図において変数判断の枝（たとえば $d_3{}^2$）の部分に適当な修正を加えれば，いくらかの状態と枝の追加で実現でき，本質的な変更を加える必要はない．

また，処理されるステートメントが算術式であることさえ判断できれば開始記号は書かなくてすむ．終了記号については2個以上続くブランクで代用できる．

---

* 各 predicate $p_i$ が表わす論理条件が満たされるとき，そのすぐ次の項 (operator または predicate) が次に実行される．論理条件がみたされないとき，右隣の上向矢印につけた番号と同一番号をもつ下向矢印へとび，その下向矢印のすぐ右の項が次に実行される．$\omega$ は偽の論理条件で，常にその右隣の上向矢印に従って無条件でとぶ．

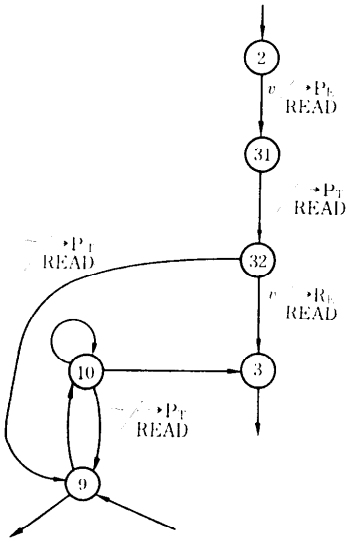


Fig. 2

以後，本論文では Fig. 1 の状態図について考える．

## 3. pds の記憶容量の評価

### 3.1 評価式

〔定義1〕 算術ステートメント $k$ のレベル $n$ は，次式で定義される．

$$n \triangleq \max_{\alpha \in S_k}\{\pi(\alpha)\} - \min_{\alpha \in S_k}\{\pi(\alpha)\} + 1 \qquad (1)$$

ただし，$S_k$ はステートメント $k$ の中に現われる演算記号の集合である．また，考えている対象が算術ステートメントだから Table 1 から $n \geqq 5$ である．

〔定義2〕 深さ $m$ のネストとは，式の中での $m$ 重の括弧を意味する．また与えられた式で最も深いネストを最大ネストという．

〔補題1〕 ステートメントに許される長さが $l$ 文字のとき，とりうる最大ネストの深さ，およびそのときの変換用 pds，実行用 pds に必要な記憶容量は，それぞれ，次式の値 $m_m, M_T^m, M_E^m$ をこえない．ただし，各ネストは有効な括弧のみからなるとする．ここで〔$a$〕は，$a$ を越えない最大の整数を意味する．

$$m_m = [l/4] - 1. \qquad (2)$$

$$M_T^m = 2m_m + 3,\quad M_E^m = m_m + 3. \qquad (3)$$

（証明） 有効な括弧のみからなり最大ネストとなりうる算術式の形は ¢ $v = v_1\alpha_1(v_2\alpha_2(\cdots(x))\cdots)$ $ である．ただし，$x$ は部分ストリングで，(1) $v_{m_m+1}\alpha_{m_m+1}v_{m_m+2}$ か，(2) $v_{m_m+1}$ のいずれかである．ゆえに $l \geqq 3+3m_m+l_0+m_m+1$．ただし $l_0$ はネストの底の文字数で3または1である．これから $m_m \leqq (l-4)/4$．また pds 容量については明らかに (1) の場合のほうが多く，ネストの底で $M_T^m = 2+2m_m+1$，$M_E^m = 1+m_m+2$ となることから式(3)が導かれる（証明終）．

〔補題2〕 ステートメントに許される長さが $l$ 文字，レベル数が $n$ で，ネストの深さが $m_m$ より小さい場合に必要な pds 容量は，次式の値 $M_T^0, M_E^0$ をこえない．ただし $l_0$ はネストの底の文字数で，式(5)をみたす奇整数である．

$$M_T^0 = (n-3)m_0 + 2 + a_0,\quad M_E^0 = M_T^0 - m_0. \qquad (4)$$

ただし，$m_0 = [(l-4)/(2l-6)]$，

$$a_0 \triangleq (l_0-1)/2,\quad 2n-6 > l_0 > 0 \qquad (5)$$

（証明） 必要な記憶容量が最大になるのは，ネストが1個で，pds へのつめ込みがネストの底に達するまで連続して行なわれる場合である．たとえば，このような場合の算術式は ¢ $x = a+b*c\uparrow(d-e*f\uparrow(\cdots(h+i/j\uparrow k))\cdots)$ $ のようになる．さて，pds へ式をつめ込む途中では $\{¢, =, )\}$ は現われないことを考えると，ネストの底の部分の文字数を $l_0$ として，$l \geqq 3+\{2\times(n-4)+1\}m_0+l_0+m_0+1$，$2(n-4)+1 \geqq l_0 > 0$．これから式(4)，(5)が補題1の場合と同様にして導かれる（証明終）．

〔定理1〕 レベル数が $n$ のステートメントに許される長さが $l$ 文字，ネストの深さが $m_R$ の時，必要とされる pds $P_T, P_E$ の記憶容量は次式の値 $M_T, M_E$ をこえない．ただし，$m \triangleq \min\{m_0, m_R\}$．

$$M_T = (n-3)m+n,\quad M_E = (n-4)m+n. \qquad (6)$$

（証明） $m' \triangleq \min\{m_m, m_R\}$ とする．また，式(4)，(5)から $a_0 \leqq n-4$ である．

〔i〕 $m_m \geqq m_0 \geqq m_R$ の時，$n \geqq 5$ およびネストの深さの制限から，$M_T^m = 2(m'+1)+1 \leqq 2(m_R+1)+1 < (n-3)m_R+n$，また $M_T^0 < (n-3)m_R+n$．$M_E^m = m'+3 \leqq m_R+3 < (n-4)m_R+n$，また $M_E^0 < (n-4)m_R+n$．ゆえに式(6)が成立する．

〔ii〕 $m_R \geqq m_m \geqq m_0$ の時，$n \geqq 5$ および式(3)，(4)，(5)から，$(M_T-2)-M_T^m \geqq M_T^0-M_T^m \geqq (n-3)\{(l-4)/(2n-6)-1\}+n-3-(2l/4) = -2$．ゆえに $M_T - M_T^m \geqq 0$，かつ $M_T - 2 \geqq M_T^0$．また，$(M_E-2)-M_E^m \geqq M_E^0 - M_E^m \geqq (n-4)\{(l-4)/(2n-6)-1\}+n-4-(l/4) \geqq -2$．ゆえに，$M_E - M_E^m \geqq 0$，かつ $M_E - 2 \geqq M_E^0$ である．

〔iii〕 $m_m \geqq m_R \geqq m_0$ のとき，$M_T^m, M_E^m$ については〔ii〕と同じ．また評価式(6)の形から明らかに $M_T^0$

$<M_T$, かつ $M_E^0<M_E$ もみたされる.

ゆえに $m$ の各値について式(6)が成立することがわかる(証明終).

### 3.2 数値例

式(6)で与えられる $M_T$, $M_E$ の評価式について具体的な数値計算の結果を **Fig. 3**, **Fig. 4** に示す. たとえば $m_R$ が 15 のとき $M_T$ は 67 語以下, $M_E$ は 52 語以下でよい.

実際に生じうる pds の必要な最大容量は, おのおの

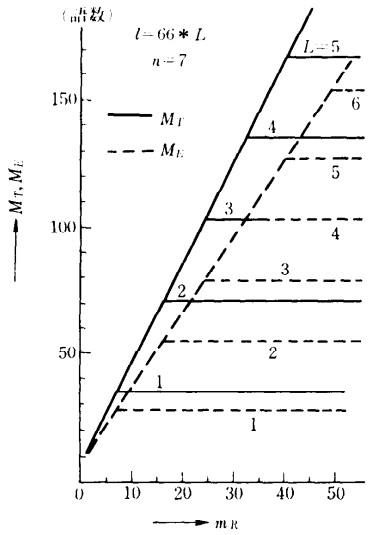


Fig. 3

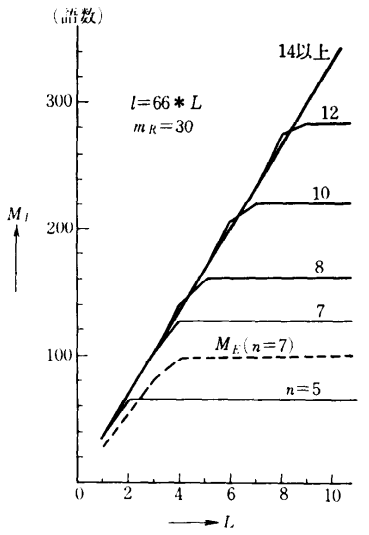


Fig. 4

$M_T'=\max\{M_T^0, M_T^m\}$, $M_E'=\max\{M_E^0, M_E^m\}$ であるが, それにくらべると式(6)の評価式で与えられる pds の容量の上界値 $M_T, M_E$ は, かなり大きめに評価されている. 実用上十分と思われる, 許されるネストの深さは約 30 であろう*. この時には補題 1, 2 および定理 1 に基ずく数値計算から次の系 1 が成立することが確かめられている.

〔系 1〕 $m_R=30$ のとき, $M_T, M_E$ は次式で与えられる値で十分である.

$$M_T=2m'+3,\quad M_E=(n-4)m+2+a_0'. \quad (7)$$

ただし, $a_0'=\{(l-2mn+6m-4)/2\}$,

$m'\triangleq\min\{m_m, m_R\}$, $m\triangleq\min\{m_0, m_R\}$.

実際には算術ステートメントで用いられる変数名, 定数は, 2 文字以上のストリングであることが多いため, ステートメントに許される長さが定まっている場合には, 実際に必要とされる記憶容量は, 定理 1, 系 1 の評価値よりかなり少なくてすむ.

## 4. 演算速度の評価

### 4.1 全処理時間と演算処理時間

本方式ではコンパイルに相当する変換が演算実行と並行して行なわれるため, 変換時間と演算実行時間とを区別して扱うよりも, 算術ステートメントの処理開始から完了までの全処理時間によって処理能率を議論するのが妥当であろう. しかし, 従来のコンパイラ方式におけるコンパイル時間に関しては一般的な評価がなされていないため, 全処理時間で両方を比較することはむずかしい. ここでは従来のコンパイラ方式の第一近似として, 原ステートメントを逆ポーランド記法による polish string へ変換し, しかるのちに演算を順次実行していく方式(直列方式とよぶ)をとり, 本装置による方式(並列方式とよぶ)と比較する.

〔定義 3〕 並列方式における全処理時間 $T_t$ は, 算術ステートメントの処理開始の時刻から処理が完了する時刻までの時間であるとする. 並列演算実行時間 $T_p$ とは, そのステートメントの処理において変換が行なわれず演算実行のみが行なわれている時間のことである. また変換時間 $T_q$ を $T_t$ と $T_p$ の差で定義する.

〔定義 4〕 直列方式での変換時間は, 原ストリングを polish string へ変換するのに要する時間とする. また, 直列演算実行時間 $T_s$ とは, polish string に従

* たとえば, 東大 HITAC 5020 E の HARP 5020 では算術式のネストの深さは 15 以下である.

って順次演算実行するのに要する時間のことをいう．

〔定義5〕 演算速度の改善度εを次式で定義する．

$$\varepsilon \triangleq T_s/T_p.$$

〔例2〕 ある演術ステートメントの処理のタイムチャートが **Fig. 5** のように与えられている時，$T_s=t_9-t_0$，$T_p=t_4-t_3+t_9-t_8$，$T_s=t_2-t_1+t_4-t_2+t_6-t_5+t_9-t_7$ である．

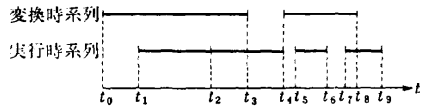


Fig. 5

〔定義6〕 加減算記号(+, −)を $\alpha_1$，乗算記号(∗)を $\alpha_2$，除算記号(／)を $\alpha_3$，代入記号(=)を $\alpha_4$ で示す時，任意の算術式について，その式の中に出現する各 $\alpha_i$ の個数を $N_i$ $(i=1,2,3,4)$ とする．

次に演算速度の評価のために基本算算の実行時間を定める．実際に使われている種々の計算機のデータをもとに一応妥当と思われる実行時間を，加減算実行時間を基準にして **Table 2** のように仮定する．乗算時間，除算時間が Table 2 の値より小さい場合には実際の改善度は以下に述べる改善度より，よくなる．

Table 2 Relative execution time of fundamental operations.

| 演 算 | 時 間 |
|---|---|
| Add, Sub, Load, Store | 1.0 |
| Mult | 5.0 |
| Div | 10.0 |
| Conditional Branch | 0.5 |
| Compare | 1.0 |
| Unconditional Branch | 0.5 |
| Shift | 2.0 |
| Indexing | 1.0 |

枝 $d_j^i$ の所要時間 $t_j^i$ とは，制御が状態 $i$ から枝 $d_j^i$ をへて状態 $j$ へ移るまでの時間を意味する．Table 2 の基本演算実行時間をもとにして求めた $t_j^i$ が，Fig. 1 の状態図の各枝 $d_j^i$ に書き込んである．

### 4.2 演算速度評価の一方法

Table 2 をもとにして演算処理時間を **Table 3** のように定める．この時間は命令解読，索表，情報転送，実行の処理を行なって，各命令を実行するのに要する時間である．ただし，評価の便宜上，巾乗演算子については考えない．

Table 3 A processing time of an operation

| $i$ | 演算記号 $\alpha_i$ | 演算処理時間 $t\alpha_i$ |
|---|---|---|
| 1 | +, − | 3.5 |
| 2 | ∗ | 7.5 |
| 3 | ／ | 12.5 |
| 4 | = | 2.5 |

Fig. 1 において演算実行開始命令のでる枝の始点である状態の集合は $R_\beta \triangleq \{17, 19, 13, 15, 6, 22, 29\}$，一時停止状態をとりうる状態の集合は $N_L \triangleq \{19, 15, 6, 7, 29, 30, 22, 24\}$ である．状態 $i \in R_\beta$ から出発して $j \in N_L$ に達するまでの最小時間 $T_j^i$（ただし，意味のあるパスについてのみ考える）が，その時実行している演算 $\alpha_k$ の処理時間 $t\alpha_k$ より大きければ，その状態 $j$ では一時停止しない．$T_j^i < t\alpha_k$ ならば，時間 $t\alpha_k - T_j^i$ だけ $j$ で一時停止する．たとえば $T_{22}^{17}=7.0$，$T_{19}^{17}=7.0$，$T_{30}^{17}=7.5$，$T_7^{17}=11.5$，$T_{19}^{13}=11.0$，$T_{24}^{22}=1.0$ である．これらを考慮して種々の基本的な算術式についての状態推移を紙上で追跡し，各 $\alpha_k$ の実行に対し $t\alpha_k$ と $T_j^i$ とを比較することから，一般に算術ステートメントを処理する時の $T_p$ の予測式が得られる．

$T_p$ の予測式の推定のためにまず括弧を含まない算術式について考察し，次に括弧を含む式を考える．

〔場合1〕 括弧を含まない算術式の場合．

式の先頭から $i$ 番目の変数を $v^i$，$j$ 番目の演算記号を $\alpha^j$ とすると，一般に算術式は次の形に書ける．

$$¢\, v^1\alpha^1 v^2\alpha^2 \cdots\cdots v^n\, \$$$

ただし $\alpha^1$ は常に = 演算である．基本的な算術式について各演算記号 $\alpha^k$ の演算実行中に変換が一時停止する時間 $\tau^k$ を求めたのが **Table 4** である．また $T_p$

Table 4 Temporary holting time of a syntax analysis. (statements containing no parentheses)

| 式番 | $\alpha^1$ | $\alpha^2$ | $\alpha^3$ | $\alpha^4$ | $\alpha^5$ | $\tau^1$ | $\tau^2$ | $\tau^3$ | $\tau^4$ | $\tau^5$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | = | + | + | + | | 0 | 0 | 0 | 0 | |
| 2 | = | ∗ | | | | 1.0 | 0.5 | | | |
| 3 | = | ∗ | ∗ | ∗ | | 1.0 | 0 | 0 | 0.5 | |
| 4 | = | ／ | | | | 1.0 | 5.5 | | | |
| 5 | = | ／ | ／ | ／ | | 1.0 | 5.0 | 5.0 | 5.5 | |
| 6 | = | + | ／ | | | 1.0 | 0 | 5.5 | | |
| 7 | = | + | ∗ | | | 1.0 | 0 | 0.5 | | |
| 8 | = | + | ∗ | + | ∗ | 1.0 | 0 | 0 | 0 | 0.5 |
| 9 | = | ∗ | ／ | | | 1.0 | 0 | 5.5 | | |
| 10 | = | ∗ | ／ | ∗ | ／ | 1.0 | 0 | 5.0 | 0 | 5.5 |
| 11 | = | ／ | ∗ | | | 1.0 | 5.0 | 0.5 | | |
| 12 | = | ／ | ∗ | ／ | ∗ | 1.0 | 5.0 | 0 | 5.0 | 0.5 |
| 13 | = | ／ | + | | | 0 | 5.0 | 0 | | |
| 14 | = | + | ∗ | ／ | | 1.0 | 0 | 0 | 5.5 | |
| 15 | = | + | ／ | ∗ | | 1.0 | 0 | 5.0 | 0.5 | |
| 16 | = 1.0 | + 0 | ∗ 1.5 | ／ 2.5 | + 0 | ∗ 1.5 | ／ 2.5 | + 0 | ∗ 0 | ／ 5.5 |
| 17 | = 1.0 | + 0 | ／ 5.0 | ∗ 0 | + 0 | ／ 5.0 | ∗ 0 | + 0 | ／ 5.0 | ∗ 0.5 |

は $\tau_i$ の和で与えられる．Table 4 において，たとえば第2式の $\tau^1$ を $\tau^1+\tau^2$ で置きかえて $\tau^2$ を0にし，また第3式の $\tau^1$ を $\tau^1+\tau^4$ で置きかえて $\tau^4$ を0とするようにして $T_p$ の値を不変に保ちつつ＊による停止時間をすべて＝，または／の停止時間に含ませると $T_p$ に関して補題3を得る．ただし Table 4～6 で，比較的演算記号数の多い式については各 $\alpha^i$ の下に $\tau^i$ が記入してある．

〔補題3〕 括弧を含まない算術式において，$T_p$ は次式の値 ${}^1T_p$ をこえない．ただし $N_4=1$.

$$ {}^1T_p=5.5N_3+1.5N_4. \qquad (8) $$

〔場合2〕 括弧を含む算術式の場合．

算術式の部分ストリングを $X$ で示す．ただし，$X$ の中には左，右括弧を含まないとする．そうすると，

（1） 代入記号と左括弧で囲まれている場合〔$=X($〕，および，2個の左括弧で囲まれている場合〔$((X($〕には，右端の左括弧が読み込まれた時点で $X$ は，$v\alpha_i$ (ただし $i=1,2$ または 3)，または $v\alpha_iu\alpha_j$ (ただし $i=1$, $j=2$ または 3) のいずれかの形になっている．

（2） 左括弧と右括弧とで囲まれている場合〔$(X)$〕は，右端の右括弧が読み込まれた時点では $v\alpha_iu$ (ただし $i=1$, $j=2$ または3)，または $v\alpha_iu\alpha_jw$ ただし $i=1$, $j=2$ または3）のいずれかの形になっている．

（3） 右括弧と左括弧で囲まれた場合〔$)X($〕には，左括弧が読み込まれた時点で $X$ は $\alpha_i$ ($i=1,2$ または3)，または $\alpha_iv\alpha_j$ ($i=1$, $j=2$ または 3) のいずれかの形になっている．

（4） 右括弧と＄で囲まれた場合〔$)X$＄〕，および2個の右括弧で囲まれた場合〔$)X))$〕は，＄または右端の右括弧を読み込んだ時点で $X$ は，$\alpha_iv$ ($i=1,2$ または3)，または $\alpha_iv\alpha_ju$ ($i=1$, $j=2$ または3) のいずれかの形になっている．

以上の（1）～（4）および〔場合1〕での考察から，可能なすべての種類の算術式を調べなくても $T_p$ の推定が可能になる．基本的な算術式に関して状態推移を紙上で追跡した結果が **Table 5～7** である．ただし，Table 5, 6 では定数，変数を省略した形で算術式は表現されている．これらの結果と補題3から次の定理を得る．

〔定理2〕 与えられた算術ステートメントにおいて，各演算記号 $\alpha_i$ ($i=1,2,3,4$) の出現する回数 $N_i$ がわかれば，その算術ステートメントの処理時の $T_p$ は，ほとんどの場合，式(8)の ${}^1T_p$ をこえることはない．ゆえに演算速度の改善度の評価式は式(9)で与えられる（これは worst case evaluation）．

$$ \varepsilon_d \wedge \frac{T_s}{T_p} \cong \frac{3.5N_1+7.5N_2+12.5N_3+2.5N_4}{5.5N_3+1.5N_4} $$

Table 5 Temporary holting time of a syntax analysis. (statements containing parentheses.－1)

| 式番 | 式の型 | $\tau^1$ | $\tau^2$ | $\tau^3$ | $\tau^4$ | $\tau^5$ |
|---|---|---|---|---|---|---|
| 18 | ＝／(＋) | 1.0 | 5.5 | 0 | | |
| 19 | ＝／(＊) | 1.0 | 5.5 | 0 | | |
| 20 | ＝／(／) | 1.0 | 8.0 | 1.5 | | |
| 21 | ＝＊(＊) | 1.0 | 0.5 | 0 | | |
| 22 | ＝＊(／) | 1.0 | 3.0 | 1.5 | | |
| 23 | ＝／(／＋) | 1.0 | 5.5 | 5.0 | 0 | |
| 24 | ＝／(／＊) | 1.0 | 5.5 | 5.0 | 0 | |
| 25 | ＝／(／／) | 1.0 | 8.0 | 5.0 | 1.5 | |
| 26 | ＝／(＊＊) | 1.0 | 5.5 | 0 | 0 | |
| 27 | ＝／(＋／) | 1.0 | 8.0 | 0 | 1.5 | |
| 28 | ＝＊(＋＋) | 1.0 | 0.5 | 0 | 0 | |
| 29 | ＝＊(＋＊) | 1.0 | 0.5 | 0 | 0 | |
| 30 | ＝＊(＋／) | 1.0 | 3.0 | 0 | 1.5 | |
| 31 | ＝＊(／＋) | 1.0 | 0.5 | 5.0 | 0 | |
| 32 | ＝＊(／＊) | 1.0 | 0.5 | 5.0 | 0 | |
| 33 | ＝／＊(＋) | 1.0 | 1.0 | 0.5 | 0 | |
| 34 | ＝／／(＋) | 1.0 | 1.0 | 5.5 | 0 | |
| 35 | ＝／／／(＋) | 1.0 | 5.0 | 1.0 | 5.5 | 0 |
| 36 | ＝／(＋＊／) | 1.0 | 8.0 | 0 | 0 | 1.5 |
| 37 | ＝＋／(＋／) | 1.0 | 0 | 8.0 | 0 | 2.5 |
| 38 | ＝＋／(／＋) | 1.0 | 0 | 5.5 | 5.0 | 0 |
| 39 | ＝ ／(＋ ＊ ／ ＋ ＊ ／ ＋ ＊ ／)<br>1.0 8.0 2.5 0 1.5 2.5 0 1.5 0 0 1.5 | | | | | |
| 40 | ＝ ／(＋ ／ ＋ ／ ＋ ／)<br>1.0 8.0 2.5 1.5 2.5 1.5 0 1.5 | | | | | |

Table 6 Temporary holting time of a syntax analysis. (statements containing parentheses.－2)

| 式番 | 式の型 | $\tau^1$ | $\tau^2$ | $\tau^3$ | $\tau^4$ | $\tau^5$ |
|---|---|---|---|---|---|---|
| 41 | ＝(＋／)／ | 1.0 | 1.0 | 0 | 5.5 | |
| 42 | ＝(＋＊)／ | 1.0 | 0 | 0 | 5.5 | |
| 43 | ＝(＋＊／)／ | 1.0 | 1.0 | 0 | 0 | 5.5 |
| 44 | ＝(＋／＊)／ | 1.0 | 0 | 5.0 | 0 | 5.5 |
| 45 | ＝／(＊／)／ | 1.0 | 7.5 | 0 | 1.5 | 5.5 |
| 46 | ＝(＋／)／(＋) | 1.0 | 0 | 0 | 5.5 | 0 |
| 47 | ＝(＋＊)／(＋) | 1.0 | 0 | 0 | 5.5 | 0 |
| 48 | ＝(＊(＋／)) | 1.0 | 8.0 | 0 | 0 | 2.5 |
| 49 | ＝ ／(／(／(／(／(＋ ／))))<br>1.0 8.0 8.0 4.0 5.0 2.0 0 2.5 | | | | | |
| 50 | ＝ ＊ ＊ ／(＊ ＊(＋ ／))<br>1.0 0 0 8.5 0 0 0 2.5 | | | | | |
| 51 | ＝ ＋ ／(＋ ／(＋ ／(＋ ／)))<br>1.0 0 8.0 0 4.0 0 5.0 0 2.5 | | | | | |
| 52 | ＝ ／(＋ ＊ ／(＋ ＊ ／(＋ ＊ ／)))<br>1.0 8.0 0 0 5.0 0 0 5.0 0 0 4.0 | | | | | |
| 53 | ＝(＋ ／)／(＋ ／)／(＋ ／)<br>1.0 0 0 3.5 0 1.5 8.0 0 1.5 | | | | | |
| 54 | ＝(＊)／(＊)／(＊)／(＊)<br>1.0 0 1.0 0 1.0 0 5.5 0 | | | | | |

Table 7 A comparison of ${}^1T_p$ evaluated by Eq. 8 and $T_p$ estimated by a simulation.

| 式番 | $N_3$ | ${}^1T_p$ | $T_p$ | 式番 | $N_3$ | ${}^1T_p$ | $T_p$ |
|---|---|---|---|---|---|---|---|
| 18 | 1 | 7.0 | 6.5 | 37 | 2 | 12.5 | 11.5 |
| 19 | 1 | 7.0 | 6.5 | 38 | 2 | 12.5 | 10.5 |
| 20 | 2 | 12.5 | 10.5 | 39 | 4 | 23.5 | 18.5 |
| 21 | 0 | 1.5 | 1.5 | 40 | 4 | 23.5 | 18.5 |
| 22 | 1 | 7.0 | 5.5 | 41 | 2 | 12.5 | 7.5 |
| 23 | 2 | 12.5 | 11.5 | 42 | 1 | 7.0 | 5.5 |
| 24 | 2 | 12.5 | 11.5 | 43 | 2 | 12.5 | 6.5 |
| 25 | 3 | 18.0 | 15.5 | 44 | 2 | 12.5 | 10.5 |
| 26 | 1 | 7.0 | 6.5 | 45 | 3 | 18.0 | 14.5 |
| 27 | 2 | 12.5 | 10.5 | 46 | 2 | 12.5 | 5.5 |
| 28 | 0 | 1.5 | 1.5 | 47 | 1 | 7.0 | 5.5 |
| 29 | 0 | 1.5 | 1.5 | 48 | 2 | 12.5 | 10.5 |
| 30 | 1 | 7.0 | 5.5 | 49 | 6 | 34.5 | 30.5 |
| 31 | 1 | 7.0 | 6.5 | 50 | 2 | 12.5 | 12.0 |
| 32 | 1 | 7.0 | 6.5 | 51 | 4 | 23.5 | 20.5 |
| 33 | 1 | 7.0 | 2.5 | 52 | 4 | 23.5 | 23.0 |
| 34 | 2 | 12.5 | 7.5 | 53 | 5 | 29.0 | 15.5 |
| 35 | 3 | 18.0 | 12.5 | 54 | 3 | 18.0 | 8.5 |
| 36 | 2 | 12.5 | 10.5 | | | | |

(9)

### 4.3 数 値 例

算術式にあらわれる各演算記号の個数と改善度の関係を **Fig. 6**, **Fig. 7** に示す．ここでは，$N_4=1$ に固定してある．また，たとえば，$N_1:N_2:N_3=100:6:2$ (ほぼ Gibson Mix 値に相当*) のときは，$\varepsilon_d \cong 30$ になる．

## 5. 本装置のシミュレーション

前節までに PADEM の基本設計とその拡張，使用する2個の pds の記憶容量の評価，演算速度の改善度の評価を行なったが，設計した PADEM によって算術ステートメントの処理が正しく行なわれること，および記憶容量の評価，改善度の評価の妥当性を確かめるために，次のようなシミュレーションを行ない，満足な結果が得られた．

### 5.1 シミュレーションプログラムの概要

処理される算術ステートメントはデータとして与えられる．主プログラムはシミュレーション部からなり，演算速度の改善度などの計算と作表はすべて副プログラムで行なわれる．主プログラムの大きさは FORTRAN ステートメント数で約600，副プログラムは

* 加減算演算の個数を #{ADD, SUB} のように書くと，Gibson Mix 値は，#{ADD, SUB, LOAD, STORE} : #{MULT} : #{DIV} = 33.0 : 0.6 : 0.2 であるから #{MULT}=6, #{DIV}=2 とすると，#{ADD, SUB}=$N_1$, #{LOAD, STORE}=$2y$ は連立方程式

$$N_1+2y=330,\quad N_1+6+2=y$$

を満たす．ゆえに $N_1=105$ となり，$N_1:N_2:N_3=105:6:2$ が Gibson Mix 相当値となる．

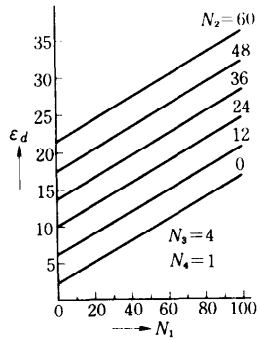


Fig. 6

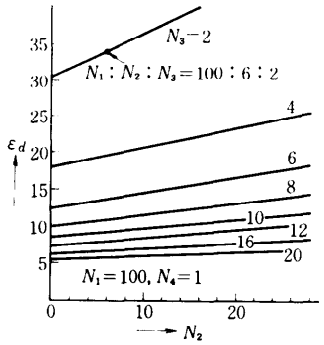


Fig. 7

16個のデックからなり約400である．コンパイル時間は約6秒，平均シミュレート時間は約1.5秒（東大大型計算機センタ HITAC 5020 E）であった．シミュレーションの結果は状態推移系列（制御情報の流れ），経過時間，演算実行状況，一時停止時間の一覧表として出力される．**Table 8** はその一例である．

### 5.2 シミュレーションに用いた算術式

実験に用いた算術式は，雑誌「情報処理」(1962～1965年) のプログラムのページに掲載された科学技術計算用プログラムから集めたもので，総数1,047個，算術式としての型の種類は53種である．その中で $y=x$ の式が全体の60%，$\{y=a\pm b,\ y=a*b,\ y=a/b\}$ の式を含めると全体の85%を占める．ここでは加算と減算の区別はせず，すべて加算として以後表現する (**Table 9**)．また，シミュレーションによって求めた改善度を $\varepsilon_s$，評価式(9)によるものを $\varepsilon_d$ で表わすことにする．

### 5.3 シミュレーションの結果

まず，シミュレーションに用いた算術式にあらわれる各演算記号数の比は，$N_1:N_2:N_3:N_4=367:219:117:1047$ である．ただし，この比は $y=x$ の型の式

Table 8 An example of a simulation result. ($V=A/B$)
Simulation of the direct execution circuit simulated stat.=1 8 2 8 7 8 1
¢ $V=A/B$ $

| PASS NODE | ($T_t$) TOTAL TIME | ($T_p$) EFF. EEX. TM. | ($T_s$) SER. SUM. TM. | LOCK TIME | LOCK NODE | 注 |
|---|---|---|---|---|---|---|
| 1 | 1.00 | 0.0 | 0.0 | | | |
| 2 | 3.50 | 0.0 | 0.0 | | | ¢→$P_T$ |
| 3 | 6.00 | 0.0 | 0.0 | | | |
| 4 | 7.50 | 0.0 | 0.0 | | | |
| 8 | 9.00 | 0.0 | 0.0 | | | |
| 9 | 12.00 | 0.0 | 0.0 | | | =→$P_T$ |
| 11 | 14.50 | 0.0 | 0.0 | | | |
| 27 | 17.00 | 0.0 | 0.0 | | | |
| 8 | 18.50 | 0.0 | 0.0 | | | |
| 9 | 21.50 | 0.0 | 0.0 | | | /→$P_T$ |
| 11 | 24.00 | 0.0 | 0.0 | | | |
| 27 | 26.50 | 0.0 | 0.0 | | | |
| 8 | 28.00 | 0.0 | 0.0 | | | |
| 16 | 29.00 | 0.0 | 0.0 | | | |
| 17 | 29.50 | 0.0 | 0.0 | | | |
| EXEC OF OPER IN PUSHT(LT) $LT$=3 IPUSHT($LT$)=7 KK=IPUSHT(LT)=7 | | | | | | $A/B$(≙$R_1$) 実行開始 |
| 8 | 30.50 | 0.0 | 12.50 | | | |
| 16 | 31.50 | 0.0 | 12.50 | | | |
| 17 | 32.00 | 0.0 | 12.50 | | | |
| 18 | 32.50 | 0.0 | 12.50 | | | |
| 8 | 34.00 | 0.0 | 12.50 | | | |
| 16 | 35.00 | 0.0 | 12.50 | | | |
| 20 | 35.50 | 0.0 | 12.50 | | | |
| 21 | 36.00 | 0.0 | 12.50 | | | |
| | | | | 5.50 | 22 | |
| EXEC OF OPER IN BUFF REGISTOR JBUFF=2 M=7 IFF 2=1 | | | | | | $V=R_1$ 実行開始 |
| 22 | 42.00 | 5.50 | 15.00 | | | |
| 23 | 42.50 | 5.50 | 15.00 | | | |
| 24 | 44.00 | 6.50 | 15.00 | | | |
| | | | | 1.00 | 24 | |
| 25 | 45.50 | 6.50 | 15.00 | | | |
| EXECUTION IS FINISHED. | | | | | | 完了 (END) |

が627個もあることの影響を強くうけている．またシミュレーションから求めた改善度 $\varepsilon_s$ を Table 9 に示す．これから一般のプログラムに現われる算術ステートメントのうちで 80% は並列演算実行時間 $T_p$ がゼロであることが知られる．さて，平均的な改善度を評価するために改善度の算術式の集合についての平均（集合平均とよぶ）$E_a(\varepsilon)$ と，考えているすべての算術式を並べた系列についての改善度の平均（系列平均とよぶ）$E_t(\varepsilon)$ を次式で定義する．ただし $\varepsilon=\varepsilon_d, \varepsilon_s$.

$$\left.\begin{aligned} E_a(\varepsilon) &\triangleq \frac{1}{N}\sum_{i=1}^{N}\frac{T_s^i}{T_p^i}=\frac{1}{N}\sum_{i=1}^{N}\varepsilon^i, \\ E_t(\varepsilon) &\triangleq \frac{\sum_{i=1}^{N} T_s^i}{\sum_{j=1}^{N} T_p^j} \end{aligned}\right\} \quad (10)$$

Table 9 The simulation result and a theoretical evaluation of $\varepsilon_d$.

| No. | 算術式の型 | 個数 | $\varepsilon_s$ | $\varepsilon_d$ |
|---|---|---|---|---|
| 1 | $v=a$ | 627 | ∞ | 1.7 |
| 2 | $v=a+b$ | 168 | ∞ | 4.0 |
| 3 | $v=a*b$ | 40 | 6.7 | 6.7 |
| 4 | $v=a/b$ | 59 | 2.3 | 2.1 |
| 5 | $v=a*b+c$ | 19 | ∞ | 9.0 |
| 6 | $v=a+b*c$ | 29 | 9.0 | 9.0 |
| 7 | $v=a/b+c$ | 1 | 3.7 | 2.6 |
| 8 | $v=a+b/c$ | 10 | 2.8 | 2.6 |
| 9 | $v=a+b+c$ | 17 | ∞ | 6.3 |
| 10 | $v=a*b/c$ | 8 | 3.5 | 3.2 |
| 11 | $v=a/b*c$ | 1 | 3.5 | 3.5 |
| 12 | $v=a+b*c*d$ | 1 | 14.0 | 14.0 |
| 13 | $v=(a+b)/c$ | 4 | 2.8 | 2.6 |
| 14 | $v=a+(b+c)$ | 1 | ∞ | 6.3 |
| 15 | $v=a*b+c*d$ | 3 | 14.0 | 14.0 |
| 16 | $v=a*(b+c)$ | 3 | 9.0 | 9.0 |
| 17 | $v=a+b/c*d$ | 2 | 4.0 | 3.7 |
| 18 | $v=a/b*c/d$ | 1 | 3.0 | 2.8 |
| 19 | $v=a/b+c/d$ | 2 | 2.7 | 2.5 |
| 20 | $v=a+b*c*d$ | 3 | 14.0 | 14.0 |
| 21 | $v=a+b*c/d$ | 2 | 4.0 | 3.7 |
| 22 | $v=a/(b*c)$ | 1 | 3.5 | 3.2 |
| 23 | $v=(a+b)*c$ | 1 | 9.0 | 9.0 |
| 24 | $v=a/(b+c)$ | 1 | 2.8 | 2.6 |
| 25 | $v=a+b*c+d*e$ | 6 | 16.3 | 16.3 |
| 26 | $v=(a+b*c)/d$ | 3 | 4.0 | 3.7 |
| 27 | $v=a+(b+c)*d$ | 1 | 11.3 | 11.3 |
| 28 | $v=a+b*(c+d)$ | 2 | 11.3 | 11.3 |
| 29 | $v=a*(b+c/d)$ | 3 | 4.7 | 3.7 |
| 30 | $v=(a+b+c)*d$ | 1 | 11.3 | 11.3 |
| 31 | $v=a*(b+c)/d$ | 1 | 4.0 | 3.7 |
| 32 | $v=(a+b)*c+d*e$ | 1 | 16.3 | 16.3 |
| 33 | $v=(a*(b+c))/d$ | 1 | 4.0 | 3.7 |
| 34 | $v=a*b+c*d+e*f$ | 3 | 21.3 | 21.3 |
| 35 | $v=a*b*c+d*e*f$ | 2 | 24.0 | 24.0 |
| 36 | $v=(a+b)/(c+d)$ | 2 | 3.4 | 3.1 |
| 37 | $v=(a*b+c*d)*e+f$ | 1 | ∞ | 21.3 |
| 38 | $v=(a+b/(c*d))*e$ | 1 | 9.6 | 4.8 |
| 39 | $v=a+b*c/d+e*f/g$ | 1 | 4.7 | 4.0 |
| 40 | $v=a*b+c*d*(e+f*g)$ | 1 | 26.3 | 26.3 |
| 41 | $v=a*b+(c+d*e*f)*g$ | 1 | 26.3 | 26.3 |
| 42 | $v=(a*b+c+d)*e/f+g$ | 1 | 8.1 | 5.8 |
| 43 | $v=(a+b)*c*d+e*f+g$ | 1 | ∞ | 23.7 |
| 44 | $v=(a+b)*c+d)*e+f$ | 1 | ∞ | 18.7 |
| 45 | $v=a+(b+c*d+e)*f/g$ | 1 | 6.2 | 5.8 |
| 46 | $v=a*b+c*(d+e/f)*g$ | 1 | 8.1 | 6.4 |
| 47 | $v=a*b+c*d*e*f+g*h$ | 1 | 31.3 | 31.3 |
| 48 | $v=(a+b)*c+(d+e)*f*g$ | 1 | 23.7 | 23.7 |
| 49 | $v=(a*b+c)*d*e+f*g+h$ | 1 | ∞ | 28.7 |
| 50 | $v=a+(b*c+\alpha+e*f)*g/h$ | 1 | 7.4 | 6.9 |
| 51 | $v=(a*(b+c*d))/(e*f)$ | 1 | 6.3 | 5.9 |
| 52 | $v=(a+b*c*d*e)/(f+g*h*i)$ | 1 | 9.2 | 8.5 |
| 53 | $v=(a+b/c*(d*e+f*g*h))/(i+j/k*l)$ | 1 | 7.0 | 4.9 |

ここで $T_p^i, T_s^i, \varepsilon^i$ は $i$ 番目の算術式の $T_p, T_s, \varepsilon$ である．また，算術式の総数を $N$(ここでは1047) とする．$T_p$ がゼロでない211個（全体の20%）について **Table 10** の結果を得た．Table 9, 10 から評価値

Table 10 Ensemble mean $E_a(\varepsilon)$ and serial mean $E_t(\varepsilon)$, evaluated by Eq. (10).

| $\varepsilon$ | $E_a(\varepsilon)$ | $E_t(\varepsilon)$ |
|---|---|---|
| $\varepsilon_s$ | 6.6 | 4.4 |
| $\varepsilon_d$ | 6.4 | 4.0 |

$\varepsilon_d$ がシミュレート値 $\varepsilon_s$ にほぼ一致していることが知られる．計算機システムとしての瞬時処理能力（1個の算術ステートメントの処理速度）の評価には集合平均 $E_a(\varepsilon)$ が，多数の算術ステートメントを逐次的に処理する場合の全体的な処理能力評価には系列平均 $E_t(\varepsilon)$ が意味をもつであろう．全処理時間 $T_t$，演算実行時間 $T_p$, $T_s$ についても1,047個の算術ステートメントに対して同様に集合平均，系列平均を定義し計算を行なった結果，$T_p$ は $T_t$ の約5％，$T_t$ は $T_s$ の約2～3倍であることが知られた．さらに，定理1で与えられる記憶容量の評価式の妥当性は Table 9 の算術式の他に，30個以上の演算記号を含む長い算術式についてのシミュレーションによっても確かめられている（**Table 11**）．

Table 11 The memory capacities $M_T$, $M_E$ of two pds's $P_T$, $P_E$.

| $m_R$ | | 0 | 1 | 2 | 9 | 19 |
|---|---|---|---|---|---|---|
| シミュレーションによる値 | $M_T$ | 4 | 7 | 7 | 31 | 60 |
| | $M_E$ | 4 | 6 | 6 | 22 | 41 |
| 評価式による値 | $M_T$ | 5 | 8 | 11 | 43 | 83 |
| | $M_E$ | 5 | 7 | 6 | 34 | 64 |

## 6. 本方式の限界と拡張

本方式は原算術ステートメントの変換と実行とを同時に行なうところに本質的な特徴があり，会話形モードで計算を行なうような時にはきわめて有効である．一方ループに含まれているような同じ型の算術ステートメントを何度も計算実行する際には，第1回目の実行時に変換された実行命令系列を記憶しておいて，2回目以後はそれを実行するようにしても，それほど演算実行時間の改善は望めないであろう．しかし，いずれにしてもこのような方式を計算機システムのハードウェアとして組み込むことにより，システムプログラムの負担をかなり軽減でき，ミニ・コンピュータや端末用計算機などに有効な方式であると思われる．たとえば，本装置を拡張したものをハードウェアに組み込むことによりコンパイラの算術ステートメント処理の手続をかなり簡単化できると考えられる．しかし，そのためにプログラミング言語とハードウェアとが密接に関係することになり，システムの融通性が低下するという欠点が現われるかもしれないが，この点はマイクロプログラム方式の積極的な利用により解決されるのではなかろうか．

さらに本方式の考え方をプログラミング言語にまで及ぼすことも考えられる．たとえば，ALGOL プログラムへの拡張は次のようにして実現できるであろう．まず，プログラムはブロック構造をしているから，begin, end を一種の括弧とみなせる．すなわち，begin が現われるとそれに続くステートメントの実行を，対応する end が現われるまで順次行なう．ステートメントの区切りを示すセミコロン（;）の演算実行優先順位を，他のいずれの記号，演算記号よりも小さく指定することによってこれが可能になる．ただし，繰り返し実行命令（for）と無条件飛びこし命令（go to）には特別の考慮が必要である．このようにして，プログラム全体が演算実行優先順位に従って完全な階層構造になっているから，プログラム全体をこの種の方式で直接実行することが可能になるであろう．

また，本方式の考え方は記号処理用インタプリタのようなソフトウエアシステムそのものの改善にも応用できるであろう．

ともあれ，このような方式を実現するためには，たとえば，プログラミング言語の構造とそのハードウェア化の方法，この種の方式をどのようにどの程度まで計算機システムに組み込むのが効果的かなど，多面的な研究が今後必要である．

## 7. おわりに

本論文では算術ステートメントをコンパイルせずに直接実行する装置の設計，その装置で用いる pds の記憶容量評価，演算速度の評価を行ない，さらに本装置のシミュレーションを行なうことにより，得られた結果の妥当性とその有効性を明らかにした．しかしながら，**6.** で述べたような限界があり，今後さらに多面的な研究が必要である．日頃熱心に討論していただく研究室の各位に感謝する．また，シミュレーションに用いたデータの収集に際して協力していただいた加藤雄一氏（現在，富士通勤務）に深謝する．

### 参考文献

1) T. R. Bashkow: A Sequential Circuit for Algebraic Statement Translation, IEEE Trans., Vol. EC-13, pp. 102～105(1964).
2) T. R. Bashkow, A. Sasson and A. Kronfield: System Design of a FORTRAN Machine, IEEE Trans., Vol. EC-16, pp. 485～499(1967).
3) J. A. N. Lee: The Anatomy of a Compiler, Reinhold Publishing Corp., pp. 162～180 (1967).
4) Yu. I. Yanov: On Logical Schemes for Algorithms, Problemy Kibernetiki, 1, pp. 75～127, Fizmatgis, Moskow (1958).
5) V. G. Lazarew: On the Synthesis of Microprogram Automata, Problemy Peredachi Informatsii, Vol. 1, No. 2, pp. 63～78(1965).