ダイナミック型内容参照記憶方式とその評価†

福永邦雄†† 笠井 保††

現在の計算機とは基本的に異なる新しいアーキテクチャをもつ計算機を構成しようとする研究が数多くなされている．本稿ではこのうち，従来の番地式記憶に代る記憶方式として，スタック形式の記憶構成を基本とする内容参照記憶方式を提案している．各記憶項目に番地付けがなされていない本記憶方式では，アクセスに際して常時記憶位置を調べる必要があるため，アクセス時間が大きくなり，またこれを改善しようとすると記憶回路が複雑になるなどの難点がある．その反面，直接情報を参照しながらアクセスする方法であるため，記憶情報を取扱う際に番地表現化する煩わしさから解放されるとともに連想機能などを容易に実現でき，柔軟な性質をもつ記憶システムが構成可能である．そこで，これらの性質を失なうことなくアクセス動作を速める方法として，記憶位置を逐次的に変化させることにより，簡単な回路構成で高速のアクセス動作が可能な記憶方式を提案し，シミュレーションによりその有効性を確めている．

1. まえがき

最近の計算機技術の発展に伴い，情報処理能力は飛躍的に増大し，その応用範囲も益々広がってきている．しかし，一方では新しい応用分野が出現するに従って，従来の計算機アーキテクチャでは処理能力を向上できない分野があることも指摘され，いわゆる非ノイマン型計算機について検討されている[1),2)]．このうち主なものは，現在の逐次処理に対して並列，連想処理[3)]，番地式記憶に対して新しい記憶方式[4)]，さらには高レベル言語処理機械[5)]，問題適応型計算機[6)]などである．

本稿では，これらの研究のうち番地式記憶に基づかない内容参照記憶方式について述べている．従来の番地式記憶においては高速ランダムアクセスが容易であるなど多くの利点を有している．しかし，記憶情報を取扱う際には必ずアドレスを介さねばならないため，基本的なプログラムはアドレス中心の記述になり，アドレス変換表を用いないと理解しにくいなど，本来記憶情報とは関係のない番地操作に大きく影響を受ける．これに対して内容参照記憶方式はアクセスに要する時間が大きくなったり，これを改善しようとすると記憶回路が複雑になる難点がある反面，アドレスに煩わされることなくアクセスできるため，記憶内容中心の取扱いが可能となる．また関連する情報すべてを取出す連想記憶，さらには記憶形態を工夫することにより記号処理をはじめとする再帰的プログラムを能率よく処理できる記憶機能など柔軟な性質をもつ記憶システムが構成可能である[4),7)]．そこで本研究では内容参照記憶の利点を失なうことなく，アクセス時間を大きく改善するため，まず記憶情報の引用状態を調べている．この結果に基づいて各記憶項目の記憶場所を実行とともに逐次移動することにより，引用される可能性の高い項目を順次アクセス時間の短い位置に配置することにより，全体としてアクセス動作の速い記憶方式を提案している[8)]．そして，本方式の性質を調べ認識処理，さらには上述の連想処理，記号処理が容易に行えることを示すとともに，そのアクセス動作の高速化をシミュレーションにより確めている[8)]．

2. 記憶情報引用状態

記憶方式について述べる前に，一般に記憶されている情報の引用についての統計的性質を調べてみる．ここでは，番地式計算機における記憶項目の引用状態と自然言語における単語の出現状態を例にとり調べてみる．

（1） 番地式計算機処理における記憶項目引用状態

現在広く用いられている番地式大型計算機においては，記憶情報の単位は数バイトからなる一語と考えて差仕えないであろう．普通，処理しようとするプログラムが与えられると，それに対応する基本的な命令列を生成し，記憶した後，順次命令を読出し実行することにより結果を得る．このとき大半の命令は一記憶単位，すなわち一語で表わされており，またプログラムに現れる各変数も通常一語で表わされ，この一語が一記憶番地に対応している．そこで記憶項目として，1つの命令または1つの変数を考え，これら記憶項目がどのように引用されるかを表1に示すフォートランで

† Dynamic Content-addressable Memory System and Its Estimation by KUNIO FUKUNAGA and TAMOTSU KASAI (Department of Electrical Engineering, Faculty of Engineering, University of Osaka Prefecture).

†† 大阪府立大学工学部電気工学教室

表 1 標本プログラムの諸性質
Table 1 Properties of sample programs.

プログラムの種類	番号	ステートメント数	変数の数	プログラムの総命令数	総記憶項目数
自然言語処理（構文解析）	1	412	17,602	1,444	19,046
ベクトル計算	2	279	17,434	830	18,264
グラフ処理	3	169	3,583	672	4,255

記述された3種類の標本プログラムについて調べる．

まず，上述の標本プログラムを実行し，記憶項目を1,000,000回引用（読出し，書込みの両操作）したとき，多く引用された記憶項目順に $e_1, e_2, \cdots, e_N$ と並べることにする．このとき，e_j の添字 j をその順位と呼び，また e_j が引用された回数 t_j をもとに，e_j の引用（出現）確率 P_j を

$$P_j = t_j/1,000,000 \qquad (1)$$

と定義する．

図1は各標本プログラムについて，引用順位とその引用確率の関係を調べ図示している．次に，総引用回数1,000,000回を n 回ごとの部分区間に分け，第 i 番目の部分区間における記憶項目 $e_{i1}, e_{i2}, \cdots, e_{iN_i}$ について，順位 j の項目 e_{ij} の引用回数 t_{ij} より

$$P_{ij} = t_{ij}/n \qquad (2)$$

なる第 i 区間における引用確率を定義する．**図2**は n=5,000 および n=50,000 としたとき，各 n についてランダムに選んだ5つの部分区間の引用順位とその引用確率の関係を調べ，その平均引用確率を示している．さらに，n 回ごとの部分区間に現れる記憶項目の種類 k_n を用いて，出現記憶項目比率 q_n を

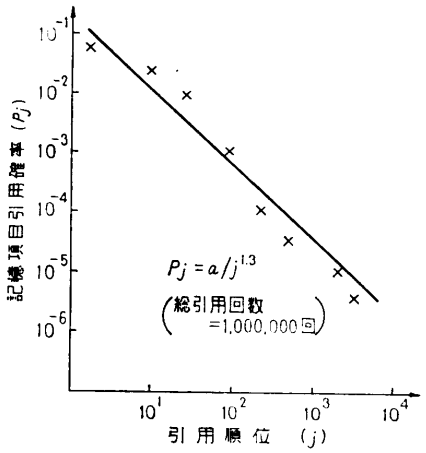

図 1 引用順位と引用確率
Fig. 1 Relation between accessed probability and order of the number of accesses.

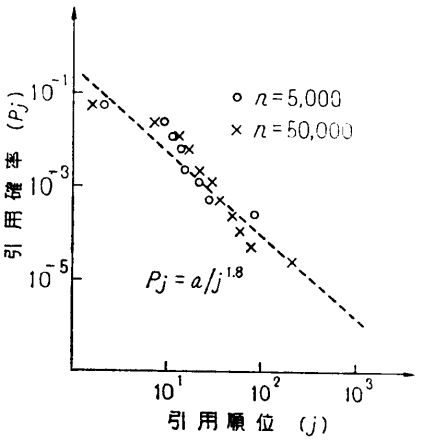

図 2 部分区間（回数 n）における引用順位と引用確率
Fig. 2 Relation between accessed probability and order of the number of accesses in a partial interval.

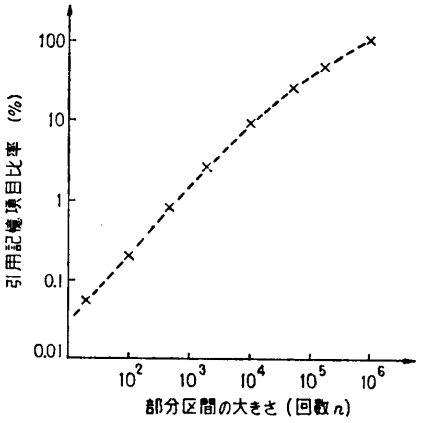

図 3 部分区間の大きさと引用項目比率
Fig. 3 Ratio of accessed items in the partial interval.

$$q_n = k_n/(\text{総記憶項目数}) \qquad (3)$$

とする．**図3**は部分区間の大きさ n とその出現比率 q_n との関係を表わしている．

（2） 自然言語における単語の出現状態

自然言語における単語は言語を認識する際の記憶情報の基本単位と考えられ，文の理解においては各単語の記憶内容を逐一引用しているとして差仕えないであろう．そこで，1つの単語に対して1つの記憶項目があるとして，その記憶項目の引用状態を調べる．ここに示す統計値は Kucera と Francis[9] が米語の15ジャンル，500トピックス（1トピックは延べ約2,000単語）の延べ1,014,232単語をもとに調べた結果である．**図4**は総文章における単語の出現回数順位とその出現確率の関係，またランダムに選んだ125トピックスに

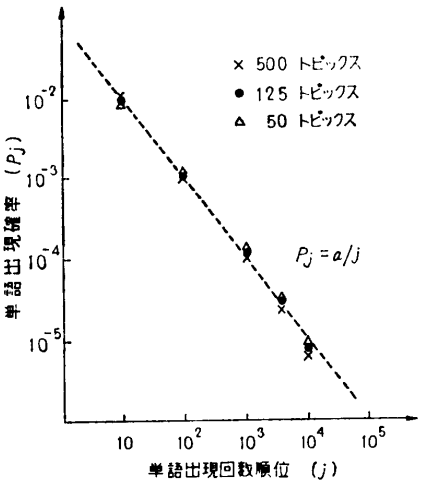

図 4 単語出現回数順位とその出現確率

Fig. 4 Relation between appeared probability and order of the number of appearances.

ついての場合，および50トピックスについての関係を図示している．

（3） 記憶情報の引用とその統計的性質

これらの結果から，記憶情報の引用状態として次のことがいえる．

（i） 各記憶項目はすべて等確率で引用されているのではなく，順位 j の記憶項目の引用確率 P_j は

$$P_j = a/j^b, \text{ ただし } a = 1/\sum_{j=1}^{N}(1/j^b) \qquad (4)$$

の関係で表わされ，パラメータ b は $1 \leq b \leq 1.3$ の範囲の値となる．図4のように b が1に等しいとき，ジップの法則[10]と呼ばれ，自然言語の単語の出現確率を表わす関係として古くから知られている．ここで得られた計算機の記憶情報の引用状態も同様な関係が成立することを示している．

（ii） 任意の小さな部分区間においても（i）と同様，各区間における引用順位と引用確率の関係もほぼ式(4)に従う．ただし，区間が異なると

$$e_{pj} \neq e_{qj} \quad (p \neq q)$$

が成立している．

（iii） 引用確率の大きい記憶項目は，あらゆる部分区間に平均して引用される傾向があり，引用確率の小さい記憶項目になるに従って，特定の部分区間に集中して引用される傾向がある．

（iv） 任意の部分区間で引用される記憶項目の種類は一般に限定されており，記憶されているすべての項目が引用されるわけではない．

これらの事実から，情報を記憶装置上に記憶する場合，記憶項目を一定の記憶場所に固定するのではなく，引用されやすい記憶項目を逐次アクセス時間が短い位置へ移動させれば，全体としてアクセス動作の高速化が期待できる．これは現在の大型計算機において記憶項目を外部記憶から主記憶，さらにはキャッシュ記憶へと移動させ，事実上 CPU とキャッシュ記憶との間で大半の情報を受渡しすることにより，最初外部記憶上にあった記憶項目をほぼキャッシュ記憶のアクセス時間で取出すことができるのは，上述の引用状態の性質に基づいていると考えられる[11),12)]．

3. 記憶情報の表現

本記憶方式を議論するための準備として，記憶情報の表現方法を明らかにする．ここで述べる記憶方式では，取扱う情報はすべて次に示す D-式 (Data-expression) の形で表現する．

〔定義 1〕

D-式は次に示す再帰的手続きにより定義する．

{D-式} ::= {ラベル}〔{S-式}〕|{ラベル}〔{D-式}〕
…{ラベル}〔{D-式}〕 (5)

{S-式} ::= {素記号}|({S-式}・{S-式})|({S-式}
…{S-式})* (6)

{ラベル} ::=〈{素記号}〉|〈{素記号}〈{ラベル}〉〉 (7)

{素記号} ::= {空}|{文字}{素記号}|{数字}{素記号} (8)

{文字} ::= $A|B|C|\cdots|X|Y|Z$ (9)

{数字} ::= $1|2|3|\cdots|8|9|0$ (10)

{空} ::= (11)

式(5)の意味は，D-式と呼ばれるものはラベルと呼ばれるものの後に，記号“〔 〕”で S-式と呼ばれるものをはさんだものか，またはラベルと呼ばれるものの後に，記号“〔 〕”で D-式と呼ばれるものをはさみ，これをいくつか並べたものである．式(6)の S-式は McCarthy が提案した S-式と同じ構造をもっている[13)]．また式(5)以下で定義される D-式は記号“〔 〕”および“〈 〉”を記号“( )”と同じものと見なせば，S-式と似た構造になる．したがって，ここで述べる記憶構造は基本的にリスト構造である．

D-式は必ず

$$D = l_1〔c_1〕\cdots l_j〔c_j〕\cdots l_n〔c_n〕 \qquad (12)$$

* S-式の表現は簡略化したリスト記法 (list notation) を用いることもある．

の形で表わされ，l_j はラベル，c_j はS-式または新たなD-式を表わしている．このとき，Dをデータと呼び，c_j をラベル l_j を持つ記憶内容と呼ぶ．

〔定義 2〕

$l_{\alpha(j_1,j_2,\cdots,j_p)}$ $(1\leq p\leq a)$ を式(7)で定義されるラベルとし，$c_{\alpha(j_1,j_2,\cdots,j_p)}$ をS-式とすると，データDが

$$\begin{aligned}D=&l_{\alpha(1)}\langle c_{\alpha(1)}\rangle\cdots l_{\alpha(j_1)}\langle l_{\alpha(j_1,1)}\langle c_{\alpha(j_1,1)}\rangle\\&\cdots l_{\alpha(j_1,j_2)}\langle l_{\alpha(j_1,j_2,1)}\langle c_{\alpha(j_1,j_2,1)}\rangle\\&\cdots l_{\alpha(j_1,j_2,j_3)}\langle\cdots\langle l_{\alpha(j_1,j_2,\cdots,j_a)}\langle c_{\alpha(j_1,j_2,\cdots,j_a)}\rangle\\&\cdots\rangle\cdots\rangle\cdots\rangle\end{aligned}\tag{13}$$

で表わされるとき，Dはa次のD-式で表わされるデータと呼び，a次であることを表わす必要のあるとき $D^{(a)}$ とする．また，$c_{\alpha(j_1,j_2,\cdots,j_p)}$ をデータDの記憶要素と呼ぶ．ただし $\alpha(j_1,j_2,\cdots,j_p)$ は正整数 $j_1,\cdots,j_p$ により決まる1つの正整数を表わし，$c_{\alpha(j_1,j_2,\cdots,j_a)}$ は最も数多く記号 "〔 〕" に包まれたS-式とする．

〔定義 3〕

u_p $(1\leq p\leq a+2)$ を素記号とすると，ラベル l が

$$l=\langle u_1\langle u_2\langle\cdots\langle u_a\rangle\cdots\rangle\rangle\rangle\tag{14}$$

と表わされるとき，a階のラベルと呼びa階であることを示す場合には $l^{(a)}$ とする．また，ラベル $l^{(a)}$ を用いて新たな $(a+2)$ 階のラベル $l_1^{(a+2)}$ を

$$l_1^{(a+2)}=\langle u_{a+1}\langle l^{(a)}\langle u_{a+2}\rangle\rangle\rangle\tag{15}$$

とすると，$l_1^{(a+2)}$ は

$$l_1^{(a+2)}=\langle u_{a+1}\langle u_1\langle u_2\langle\cdots\langle u_a\langle u_{a+2}\rangle\rangle\cdots\rangle\rangle\rangle\tag{16}$$

で与えられるものと定義する．

〔定義 4〕

式(13)のD-式から記憶要素 $c_{\alpha(j_1,j_2,\cdots,j_p)}$ を単独に取出したとき，この記憶要素はラベル

$$l=\langle l_{\alpha(j_1)}\langle l_{\alpha(j_1,j_2)}\langle\cdots\langle l_{\alpha(j_1,j_2,\cdots,j_p)}\rangle\cdots\rangle\rangle\rangle\tag{17}$$

を持つと定義し，このラベルを用いて

$$e_{\alpha(j_1,j_2,\cdots,j_p)^p}=l\langle c_{\alpha(j_1,j_2,\cdots,j_p)}\rangle\tag{18}$$

を定義する．このとき，$e_{\alpha(j_1,j_2,\cdots,j_p)^p}$ はデータDのもとでの記憶要素 $c_{\alpha(j_1,j_2,\cdots,j_p)}$ の記憶項目と呼ぶ．ただし，右肩添字pは記憶要素 $e_{\alpha(j_1,j_2,\cdots,j_p)}$ がD-式の中でp重の "〔 〕" に包まれていることを示す．

〔定義 5〕

式(13)で与えられるデータDのもとでの各記憶項目を用いて一次のD-式 $\tilde{D}$ を

$$\tilde{D}=e_{\alpha(1)^1}\cdots e_{\alpha(j_1,1)^2}\cdots e_{\alpha(j_1,j_2,1)^3}\cdots e_{\alpha(j_1,j_2,\cdots,j_a)^a}\tag{19}$$

で定義したとき，$\tilde{D}$ をデータ D の記憶項目表現と呼ぶ．

〔定義 6〕

あるデータ D_1 の記憶要素 c_j の記憶項目 $e_j{}^{\beta_j}$ が

$$e_j{}^{\beta_j}=l^{(\beta_j)}\langle c_j\rangle\quad(1\leq j\leq n)\tag{20}$$

と表わされるとき，D_1 の記憶項目表現 $\tilde{D}_1$ が

$$\tilde{D}_1=e_1{}^{\beta_1}e_2{}^{\beta_2}\cdots e_j{}^{\beta_j}\cdots e_n{}^{\beta_n}\tag{21}$$

ただし $\beta_j\leq\beta_{j+1}$ $(j=1,2,\cdots,n-1)$

と表わされ，かつ記号 "〈 〉" および "〔 〕" の数の和が最小になるD-式をデータ D_1 の最簡表現と呼ぶ．

(例 1) D-式の例として，英単語 drop を説明するD-式を考えよう．

$$\begin{aligned}D=&\langle\text{NOUN}\rangle\langle(\text{SHIZUKU TENTEKI})\rangle\\&\langle\text{VERB}\rangle\langle\langle\text{FORM}\rangle\langle\langle\text{PAST}\rangle\\&\langle\text{DROPPED}\rangle\langle\text{ING}\rangle\langle\text{DROPPING}\rangle\rangle\langle\text{VI}\rangle\\&\langle(\text{SHITARU OCHIRU})\rangle\\&\langle\text{VT}\rangle\langle(\text{SHITARASU OTOSU})\rangle\rangle\rangle\end{aligned}$$

このとき，データDの記憶要素は

c_1=(SHIZUKU TENTEKI)
c_2=(SHITARU OCHIRU)
c_3=(SHITARASU OTOSU)
c_4=DROPPED
c_5=DROPPING

であり，記憶項目表現 $\tilde{D}$ は

$$\tilde{D}=e_1{}^1e_4{}^3e_5{}^3e_2{}^2e_3{}^2$$

ただし
$e_1{}^1=\langle\text{NOUN}\rangle\langle(\text{SHIZUKU TENTEKI})\rangle$
$e_2{}^2=\langle\text{VERB}\langle\text{VI}\rangle\rangle\langle(\text{SHITARU OCHIRU})\rangle$
$\vdots$
$e_5{}^3=\langle\text{VERB}\langle\text{FORM}\langle\text{ING}\rangle\rangle\rangle\langle\text{DROPPING}\rangle$

となる．またデータ D の最簡表現 D_s は

$$\tilde{D}_s=e_1{}^1e_2{}^2e_3{}^2e_4{}^3e_5{}^3$$

で表わされるD-式のうち，記号 "〈 〉" および "〔 〕" の和が最小になるD-式であるので

$$\begin{aligned}D_s=&\langle\text{NOUN}\rangle\langle(\text{SHIZUKU TENTEKI})\rangle\\&\langle\text{VERB}\rangle\langle\langle\text{VI}\rangle\\&\langle(\text{SHITARU OCHIRU})\rangle\langle\text{VT}\rangle\\&\langle(\text{SHITARASU OTOSU})\rangle\\&\langle\text{FORM}\rangle\langle\langle\text{PAST}\rangle\langle\text{DROPPED}\rangle\langle\text{ING}\rangle\\&\langle\text{DROPPING}\rangle\rangle\rangle\end{aligned}$$

が最簡表現の1つである．

(例 2) 番地式記憶方式で記憶されている情報をD-式で表現することを考えよう．この場合，各記憶要素 c_j $(1\leq j\leq n)$ には必ず番地 a_j が割当ててあるため，各記憶項目 e_j を

$$e_j=\langle a_j\rangle\langle c_j\rangle\quad(1\leq j\leq n)$$

とし，記憶情報全体 D を

$$D=\langle a_1\rangle\{c_1\}\langle a_2\rangle\{c_2\}\cdots\langle a_n\rangle\{c_n\}$$

と表わせば，すべての情報を番地式記憶と同様に取扱うことができる．

4. ダイナミック形記憶方式

ここで提案する記憶方式は記憶しようとする情報をすべてD-式の形で表わし，これを番地式記憶装置上に記憶するのではなく，直接D-式の形で記憶する方法である．いま記憶しようとする情報がデータDとすると，その記憶項目表現 $\tilde{D}$ が

$$\tilde{D}=e_1^1 e_2^1\cdots e_{n_1}^1 e_{n_1+1}^2\cdots e_j^{\beta_j}\cdots e_n^{\beta_n} \qquad (22)$$

ただし　$\beta_j\leq\beta_{j+1}$　$(j=1,2,\cdots,n-1)$

と表わされ，また

$$D=D_1D_2 \qquad (23)$$

ただし　$\tilde{D}_1=e_1^1 e_2^1\cdots e_{n_1}^1$

$\tilde{D}_2=e_{n_1+1}^2\cdots e_j^{\beta_j}\cdots e_n^{\beta_n}$

とすると D_2 はいつも最簡表現になっているD-式の形で記憶するものとする．

（1） 記憶情報の動的移動

本記憶方式のように無番地記憶方式では，必要な情報をアクセスするとき，あらかじめその記憶位置がわからないため，必要とする情報のラベルまたは記憶要素をもとに取り出さねばならない．そこで，前述の記憶情報引用状態を参考にして，次のように情報を逐次移動することにより，アクセス時間の短縮を図る．

（i） あらゆる部分区間で引用確率が高い記憶要素は常時アクセス時間の短い位置に記憶する．

（ii） 引用確率の低い記憶要素は特定の部分区間に集中して出現する傾向があることから，この種の記憶要素は同一部分区間で引用されやすい記憶要素ほど，記憶位置が近くなるように逐次配置し，記憶領域が大きいがアクセスに要する時間が幾分多くかかる位置に記憶しておく．そして一度これらの1つが引用されると，その周辺にある記憶要素をも含めてアクセス時間が短い位置に移動することにより，高速アクセス可能な記憶領域を有効に利用し，全体としてアクセス時間を短くする．

これらの動作を複雑な演算を要することなく行う情報移動法として，次に示す方法を考える．

（2） アクセスと記憶情報動的移動

いま，m回記憶情報をアクセスした後，記憶装置上に記憶情報 D_m をD-式の形で記憶されており，その記憶項目表現 $\tilde{D}_m$ は

$$\tilde{D}_m=e_1^1 e_2^1\cdots e_{n_1}^1 e_{n_1+1}^2\cdots e_p^{\beta_p}\cdots e_n^{\beta_n} \qquad (24)$$

で表わされるとする．もちろん式(23)のところで述べたように D_m の2次以上の項は最簡表現になっている．つぎに，$(m+1)$回目に記憶項目 $e_p^{\beta_p}$ が引用されたとすると，記憶情報 D_m を次に示す方法により D_{m+1} の形に変更する．

（a） $e_p^{\beta_p}$ が引用されると，pに対して

$$p'=\begin{cases}\{p\cdot\lambda(p)\}_G+1; & \beta_p=1\\ \{n_1\cdot\lambda(p)\}_G+1; & \beta_p\geq 2\end{cases} \qquad (25)$$

$$s_j=\{p\cdot\delta_j(p)\}_G \qquad (j=1,2)$$

を求める．ただし記号 “$\{\ \}_G$” はガウスの記号を表わし，pの関数 $\lambda(p)$, $\delta(p)$ は

$$0\leq\lambda(p)\leq 1,\quad 0\leq\delta_j(p)\leq 1$$

を満たす関数である．

（b） 記憶要素 $e_p^{\beta_p}$ を e_p^1 に変更して $\tilde{D}_m$ の先頭に移動する．

（c） $e_{p-s_1}^{\beta_p-s_1}\cdots e_{p-1}^{\beta_p-1}e_{p+1}^{\beta_p+1}\cdots e_{p+s_2}^{\beta_p+s_2}$ を $e_{p-s_1}^1\cdots e_{p-1}^1 e_{p+1}^1\cdots e_{p+s_2}^1$ に変更し，この部分を $\tilde{D}_m$ の p' 番目の位置に移動する．したがって，

$$\begin{aligned}\tilde{D}_{m+1}=&e_{p'}^1 e_1^1 e_2^1\cdots e_{p'-1}^1 e_{p-s_1}^1\cdots e_{p-1}^1 e_{p+1}^1\\ &\cdots e_{p+s_2}^1 e_{p'}^1 e_{p'+1}^1\cdots e_{n_1}^1 e_{n_1+1}^2\\ &\cdots e_{p-s_1-1}^{\beta_p-s_1-1} e_{p+s_2+1}^{\beta_p+s_2+1}\cdots e_n^{\beta_n}\end{aligned} \qquad (26)$$

と表わされる記憶項目表現をもつD-式，$\tilde{D}_{m+1}$ を新たな記憶情報とする．

また新たに記憶項目 e_b が記憶されたとき，

（a） D_m に対して

$$\tilde{D}_{m+1}=e_b^1\tilde{D}_m \qquad (27)$$

となるD-式，D_{m+1} に変更する．

ここに示した移動法の意味は $(m+1)$ 回目で引用された記憶項目 $e_p^{\beta_p}$ を e_p^1 に変更し，先頭に移すことにより，一度引用された項目は時間的に近い時点で再度引用されやすい性質に備えている．また $e_p^{\beta_p}$ の近くに位置していた記憶項目 $e_{p-s_1}^{\beta_p-s_1}\cdots e_{p+s_2}^{\beta_p+s_2}$ も p' の位置に移動したのは前述の（ii）の条件に備えるためであり，β_p が大きい値のときは最簡表現の性質により，ラベルに共通した素記号を多く有していることにより，同一時点で引用される性格の強い記憶項目であることに基づいている．

なお，式(25)で定義した関数 $\lambda(p)$ および $\delta_j(p)$ の値により，次のように分類する．

一般移動法
- 定率移動法 $(\lambda(p)=\text{const})$
  - 定率定群移動法 $(\delta_j(p)=\text{const})$
  - 単純移動法 $(\delta_j(p)=0)$
- 定群移動法 $(\delta_j(p)=\text{const})$
  - 定群先頭移動法 $(\lambda(p)=0)$
  - 単純移動法 $(\delta_j(p)=0)$

(3) 記憶情報のアクセス

情報の読出しと書込みに分けて考えよう.

(i) 情報の読出し

情報の読出し方法は大きく分けて2通りある. 1つは記憶項目のラベルをもとに記憶要素を読出す方法であり, もう1つは記憶要素をもとにそのラベルを読出す方法である.

(a) ラベル $l^{(p)}$ をもとに, このラベルを持つ記憶要素を読出す場合には装置上の記憶項目 $e_j\beta_j=l_j\langle c_j\rangle$ のラベル l_j を $j=1$ から順次調べ, $l^{(p)}$ と一致したとき $(j=k)$ の記憶要素 c_k を読出した後, 前述の記憶移動を行う.

(b) 記憶要素 c のラベルを読出す場合も同様に, 記憶項目 $e_j\beta_j=l_j\langle c_j\rangle$ の記憶要素 c_j を調べ, c_j と一致したとき $(j=q)$ のラベルを l_q 読出し, 前と同様記憶移動を行う.

(ii) 情報の書込み

(a) すでに記憶されている記憶項目 $e=l\langle c\rangle$ の記憶要素を c' (またはラベルを l') に変更するとき, 上記(a)(または(b))と同様にして c_k (または l_q) を c' (または l') に変更するとともに記憶移動を行う.

(b) 新たに記憶項目 $e=l\langle c\rangle$ をつけ加える場合には, 式(27)のところで述べた記憶変更を行う.

(4) 記憶情報の整理

記憶情報を一つ整理するとは, データ D_m の記憶項目表現 $\tilde{D}_m$ が式(24)で表わされているとき, $e^1{}_{n_1}=l^{(\beta_{n_1})}\langle c_{n_1}\rangle$ を

(i) $\beta_{n_1}\geq 2$ のとき, $e_{n_1}{}^{\beta_{n_1}}$ に変更し, D_m の2次以上の項を最簡表現することである.

(ii) $\beta_{n_1}=1$ のとき, $e_{n_1-1}{}^1, e_{n_1-2}{}^1\cdots$ を順次調べ, ラベルが2階以上になる項について(i)の操作を行う.

なお, 記憶情報の整理は常時行うのではなく, 必要が生じた時点で行うものとする.

5. 記憶回路構成

本記憶方式は両方向シフトレジスタで構成されるスタックを用いた記憶装置で実現できる. すなわち, 図5に示すように大きく分けて3つのスタックと比較回路である. 比較回路は一部情報を記憶するとともに, アクセスする際にラベルの一致または記憶要素の一致を調べる回路が組込まれている.

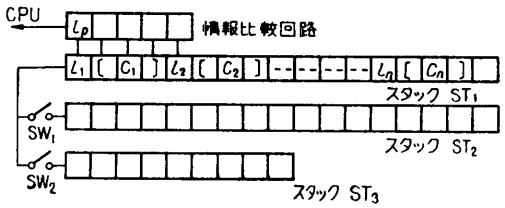

図5 スタック(両方向シフトレジスタ)による構成

Fig. 5 Constitution of memory system with stacks (sift-resisters).

次に, この記憶装置のアクセス動作について述べよう. ここでは説明の簡単化のため, 記憶情報はすべて一次のD-式で表わされていると仮定する. いま, m 回のアクセスの後, スタック ST_1 上には記憶情報 D_m

$$D_m=e_1e_2\cdots e_n$$

ただし $e_j=l_j\langle c_j\rangle$

が記憶されているとする. この情報の中からラベル l_p をもつ記憶要素 c_p を読出す操作は次の手順で行う.

(a) SW_1, SW_2 を閉じ, ST_1 の内容をポップアップ (pop up) するとともに, ST_2, ST_3 にプッシュダウン (push down) しながら, ラベル l_p を持つ記憶項目を比較回路で調べ, 見つかるまで続ける. l_p が検出できたとき, 各スタックは

$$ST_1=e_pe_{p+1}\cdots e_n$$
$$ST_2=e_{p-1}e_{p-2}\cdots e_1$$
$$ST_3=e_{p-1}e_{p-2}\cdots e_{p-k}$$

が記憶されている.

(b) この時点で記憶要素 c_p を読出し, e_p を比較回路で記憶するとともに, SW_1 を開き ST_2 の内容をポップアップし, SW_2 は閉じたまま ST_1 の内容を ST_3 にシフトさせ

$$ST_1=e_{p+s_2+1}\cdots e_n$$
$$ST_2=e_{p-s_1-1}\cdots e_1$$
$$ST_3=e_{p+s_2}e_{p+s_2-1}\cdots e_{p+1}e_{p-1}e_{p-2}\cdots e_{p-k'}$$

とする.

(c) SW_1 を閉じ, ST_2 の内容のうち $e_{p-s_1-1}e_{p-s_1-2}\cdots e_{p'}$ を ST_1 にシフトさせ

$$ST_1=e_{p'}e_{p'+1}\cdots e_{p-s_1-1}e_{p+s_2+1}\cdots e_n$$
$$ST_2=e_{p'-1}e_{p'-2}\cdots e_1$$
$$ST_3=e_{p+s_2}\cdots e_{p+1}e_{p-1}\cdots e_{p-k'}$$

に変化させる.

(d) SW_2 を閉じ，ST_3 の内容 $e_{p+s_2}\cdots e_{p+1}e_{p-1}\cdots e_{p-s_1}$ を ST_1 にシフトさせ

$ST_1=e_{p-s_1}e_{p-s_1+1}\cdots e_{p-1}e_{p+1}\cdots e_{p+s_2}e_{p'}e_{p'+1}\cdots e_n$

$ST_2=e_{p'-1}e_{p'-2}\cdots e_1$

$ST_3=e_{p-s_1-1}e_{p-s_1-2}\cdots e_{p-k'}$

とする．

(e) 最後に，ST_2 の内容を ST_1 にシフトさせ，ST_1 に e_p をつけ加えるとともに，ST_3 を空にする．

$ST_1=e_pe_1e_2\cdots e_{p'-1}e_{p-s_1}e_{p-s_1+1}\cdots e_{p-1}e_{p+1}$
$\cdots e_{p+s_2}e_{p'}\cdots e_n$

ST_2=空

ST_3=空

以上の操作により，一つの記憶情報のアクセスサイクルを終り，次の情報のアクセスに備える．なお，ここでは記憶内容が一次のD-式で表わされている場合を述べたが，一般にk次のD-式で表わされる記憶内容についても，比較回路に〔定義 4〕で述べた記憶項目に変換する操作回路をつけ加え，必要な時点で記憶項目の形で表わせば同様に取扱うことができる．

6. シミュレーション

本記憶方式のアクセス時間を調べるために，計算機によるシミュレーションを行った．対象とした標本プログラムは表1に示した3つのプログラムである．これらについて，もう少し詳しい説明をつけ加えておこう．プログラム1は入力データとして英文を読取り，単語を単位として構文解析を行うプログラムである．入力英文は10単語前後からできており，順次文を読込み解析結果を出力する．プログラム2は3次元空間に適当に散らばった約3,000のベクトルの中から，新たに入力されたベクトルとの距離が最小になるベクトルを探出すものであり，プログラム3は節点数nのグラフのハミルトンループを見い出すプログラムである．前にも述べたように，これらはいずれもフォートランで記述されている．

シミュレーションの準備として，まずプログラムの各ステートメントを，ACOS-6の汎用マイクロアセンブルプログラム (GMAP) で表現[14]しなおし，対応する命令を見い出す．これらの命令に対してプログラムの最初から順番に番号をつけていき，これが終ると変数（配列変数の場合は配列要素）に対しても引続き番号をつけていく．これら各命令または各変数を記憶要素とし，それに対応する番号をラベルとする記憶項目を考え，これら全体をプログラムに対する記憶情報として，例2のように一次のD-式で表わす．シミュレーションプログラムは標本プログラムの各ステートメントの前または後に，そのステートメントを実行するのに必要な命令番号と変数番号を引数とする．記憶装置シミュレートサブルーチン (MSS) を呼ぶ CALL 文を挿入して作成する．MSS では3つのスタックと情報比較回路をプログラム上で構成し，記憶情報移動法として定率定群移動法を用いる．このとき，パラメータ $\lambda(p)$, $\delta_1(p)$ は

$$\lambda(p)=\delta_1(p)=c \text{ (定数)}, \quad \delta_2(p)=0 \tag{28}$$

とした．このプログラムを実行させることによりシミュレーションを行う．本記憶方式のアクセス時間を測る単位として，スタック ST_1 上にある．いま呼出そうとする記憶項目がシフトさせ始めてから取り出し位置（情報比較回路部分）にn番目に現れたとき，n単位時間要するとした．シミュレーションでは，プログラム1および2では50万回，プログラム3では100万回のアクセスを行い，その平均値をアクセス時間とした．**図6**は式(28)の定数cを数種類変化させたときのアクセス時間を示している．なお，$\lambda(p)$ と $\delta_1(p)$ を違った値にしてもアクセス時間は余り短縮されず，移動させる位置と項目数は正比例しているのがほぼ最適であったため，式(28)の関係とした．また $\delta_2(p)=0$ としたのは，$\delta_2(p)$ を大きく選んでも結果に良い影響を与えなかったためである．

実際のアクセス動作に要する時間をシミュレーションプログラムから調べるには，本プログラムが標本プログラム自体の演算時間を含んでいること，またシミュレーションをすべてソフトで行っているため，

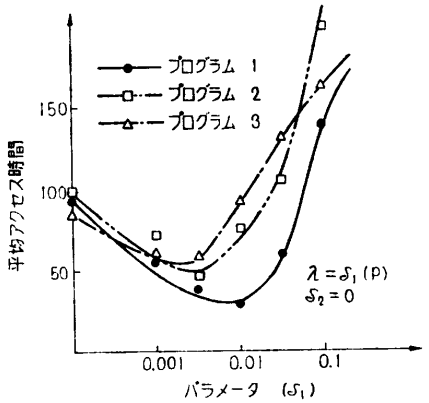

図 6 各標本プログラムにおける平均アクセス時間

Fig. 6 Mean access time for sample programs.

表 2 動的移動法（本方法）による平均アクセス時間の改善

Table 2 Improvement of mean access time using dynamic rearrangement system.

プログラムの種類	平均アクセス時間	
	記憶位置を固定	動的移動法*
プログラム 1	7,808	37.8
プログラム 2	12,817	47.0
プログラム 3	2,972	58.0

* $\lambda(p)=\delta_1(p)=0.0033$

ハードウェアで構成することを前提にすると余分な処理時間が大きく加わることを考慮せねばならないため，推定不可能である．しかし，一応の目安としてシミュレーションに要した時間を示すと，ACOS Series 77 Model 600 ではプログラム 1, 2 の 50 万回のアクセスに対して各々約 1,500 秒，プログラム 3 では 100 万回に対して約 1,800 秒であった．

以上の結果から，本記憶方式の定率定群移動法においては $\lambda(p)=\delta_1(p)=0.005$ 付近でアクセス時間が最小になる．これを記憶情報移動を行わない，すなわち実行開始時の記憶位置に固定したままの状態で記憶項目を引用した場合のアクセス時間と比較すると**表 2** のようになり，改善比は数十倍から 200 倍以上にもおよび，本方法の有効性が確められる．

7. 考　察

本記憶方式の特徴をまとめてみよう．

（1） 現在多く用いられている番地式記憶方式と比較すると**表 3** の通りになる．アクセス方法，リスト処理については，これまでに述べたことにより容易に理解できる．つぎに，本方式はパターン認識など認識処理に向いている．これは記憶項目のラベルを入力パターン信号そのものとし，記憶要素にその意味を割当てれば認識処理の際，入力未知パターンと記憶されているパターンとの同定操作において番地情報を介さずに処理でき，また直接連想記憶なども容易に行える．さらに，認識しようとするパターンが，その標準パターンに近い程，発生頻度が高いと考えられるため，記憶情報移動により，アクセス時間が短くなり，認識時間が少なくてすむ．一方，標準パターンから隔った特殊なパターンは，前とは逆に認識時間は多くかかるが，一度記憶しておけば呼出すことができるなど，認識処理に適している．

表 3 番地式記憶方式との比較

Table 3 Comparison of the proposed memory system with address-memory systems.

本方式	従来の記憶方式
内容参照記憶	番地式記憶
順アクセス	ランダムアクセス
連想記憶容易	連想記憶不向き
記号処理，認識処理	数値データ処理
構造化情報記憶	平面的情報記憶

（2） 本記憶装置を構成するとき，図 5 の回路構成を並列に多数配置すれば，並列アクセス処理が可能になり，さらにアクセス時間を短縮できる．

（3） 本方式においては，各記憶項目のラベルまたは記憶内容の記号列長の長短が，アクセス時間の速さに影響する．そのため，よく引用される記憶項目の素記号はなるべく短い記号列で表現し，引用確率の低い項目に対しては長い記号長の素記号で表わす最適符号化法[15]を用いれば，さらにアクセス動作を速めることができる．自然言語においては，一般に高出現頻度の単語ほど記号長が短いことはよく知られており，情報伝達の面だけからではなく，記憶形態からも記号長が決まっていると考えることができれば，人間の記憶と類似性があり興味深い．

（4） シフトレジスタ形の記憶回路を集積化することは容易であり，実用化に際しての問題点は少ない．

8. むすび

番地式記憶に基づかない記憶方式として，ダイナミック形内容参照記憶方式を提案した．この方式は記号処理，認識処理などに適した方法であり，また直接記憶情報を参照しながらアクセスする方法であるため，連想記憶機能など番地式記憶にない柔軟な性質をもつ記憶が実現できる．従来，この種の記憶方式ではアクセス時間が大きくなるか，またこれを改善しようとすると記憶回路が複雑になるなどの難点があるが，ここでは比較的簡単な回路構成で，アクセス時間を短くできることを示した．なお残された問題として，本記憶方式に対応するプログラム言語を考えること，また連想記憶，記号処理さらには認識処理などへの応用を考えることである．

謝辞 本研究に際しシミュレーションの一部を実行していただいた本学院生辻井建八君（現在日本電気(株)）に感謝致します．なお本研究は松永記念科学振興財団の奨励研究であることを付記する．

参考文献

1) Backus, J.: Can Programming Be Liberated from the Von Neumann Style? A Functional Style and Its Algebra of Programs, CACM, Vol. 21, No. 8 (1978).
2) 相磯秀夫他：新しい計算機アーキテクチャ-非ノイマン機能，情報処理，Vol. 19, No. 12 (1978).
3) 村岡洋一：並列処理アーキテクチャ，情報処理，Vol. 19, No. 12 (1978).
4) Yau, S. S. et al.: Associative Processor Architecture, A Survey, ACM Computing Surveys, Vol. 9, No. 1 (1977).
5) 箱崎勝也：高級言語マシンのアーキテクチャ，情報処理，Vol. 19, No. 12 (1978).
6) 坂村，相磯：計算機アーキテクチャの自動最適化に関する考察，信学論，Vol. 60-D, No. 11 (1977).
7) Parhami, B.: Associative Memories and Processors, An Overview and Selected Bibliography, Proc. IEEE, Vol. 61, No. 6 (1973).
8) 福永，笠井：ラベル型ダイナミック記憶とその性質，信学技報，EC 79-7 (1979).
9) Kucera, H. and Francis, W. N.: Computational Analysis of Present-Day American English, Brown Univ. Press, Providence (1967).
10) 細井 勉：暗号と字引，数理科学，1966, 1 (1966).
11) 飯塚 肇：キャッシュ・メモリ・システム，情報処理，Vol. 13 (1973).
12) Fukunaga, K. and Kasai, T.: The Efficient Use of Buffer Storage, Proc. ACM 77 Conf. (1977).
13) McCarthy, J.: Recursive Functions of Symbolic Expressions and their Computation by Machine, CACM, Vol. 3 (1960).
14) ACOS-6 GMAP 説明書，日電東芝情報システム(株) (1979).
15) 笠原芳郎：情報理論と通信方式，共立出版(1969).

(昭和54年8月30日受付)
(昭和55年7月17日採録)