新しいアドレスパターン発生方式による キャッシュ・メモリのシミュレーション*

飯塚　肇** 照井武彦***

Abstract

This paper describes a new method of program address pattern generation and some interesting results of cache memory simulation which uses this techique.

This address generation method is based on the utilization of the working set concept to represent address locality and can produce the patterns which are very similar to practical ones. It also has advantages of the flexibility, speed, easy to program etc.

The cache memory simulation reported in this paper is mainly aimed to the comparisons of various mapping and storing methods, and the study of the effect of program switching. Some of the results obtaind are the effective cycle time, block life, block utilization efficiency, miss rate etc. In particular, from the simulation results we might conclude th set associative mapping is most effective.

1. まえがき

キャッシュ・メモリ方式は，2種の主記憶媒体間の性能と価格の差，およびプログラムアドレスパターンの強いローカル性に着目して，1968年実用化された実効的に高速で，かつ大容量の主記憶システムを実現する技法である[1]．この技法の今後の成行きに関しては，一概に結論を出すことはできないが，現在は大型機においてかなり広く利用されているので，その性能の評価は個々のシステムの設計パラメータの決定のためにも，その有効性の判断のためにも極めて重要な問題である．

評価の手法としては解析的計算[2]とシミュレーションによるものがある．前者は簡単ではあるが，精度が上らず，細部の状況を評価するのには向かないので，シミュレーションによる方法がよく用いられ，二，三の報告[3~5]もあるが，それらはいずれも，実際のプログラムをトレースして得たアドレスパターンを逐一シミュレーションしていく方法であって，トレースされたプログラムを走らせた場合に関しては100%正しい結果が得られる反面，トレースによるデータ収集とプログラム開発に多大の時間と費用を要し，また，タイミングが関連する場合や，マルチプログラム等の特殊情況を発生して，その影響を測定することは極めて困難である．

我々はそれに代え，プログラムをシミュレーション言語としての機能がととのっている GPSS で行なうこととし，アドレスパターンについても乱数をもとに計算機内で現実のパターンに近いものを簡単に発生できる方式を考案し，それを用いて，シミュレーションを行ない[6]，マッピング方式†間の性能差，直接ストア†とスワップ形†の性能差，ブロックの寿命分布，マルチプログラミング時における有効性等，従来のシミュレーションであまり測定されなかったデータをも得ることができた．以下，この新しいアドレスパターンの発生方式（これは先行制御の効果等アドレスパター

* Cache memory simulation by a new method of address pattern generation, by Hajime IIZUKA (Electrotechnical Laboratory) and Takehiko TERUI (University of Iwate)

** 電子技術総合研究所電子計算機部
*** 岩手大学工学部電子工学科

† キャッシュを N 個のブロックに分割する時，主記憶の各ブロックが割当を受けることのできるブロック数を k とすれば，$k=1$ の時直接形マッピング，$k=N$ の時完全アソシアティブ，$k=m$ $(2\leq m\leq(N/2))$ の時セット内ブロック数 m のセットアソシアティブ形と呼ぶ．更に，転送ブロックとマッピングブロックを別にし，一つのマッピングブロック中に複数の転送ブロックを含め $k=N$ のまま N の数を小さくしたものをセクタ形という．

またストア方式の直接ストアとは記憶の書き換えが起った時，直ちに主記憶も書きかえる方法をさし，スワップ形はその場ではキャッシュのみを書き換え，割り当て変更時に主記憶を書き換えるものをさす．これらの詳細は文献1）等を参照のこと．

ンの関係するシミュレーションに一般的に応用できるものである.)とそれを用いにキャッシュ・メモリのシミュレーションの結果例について，その要点を述べるが，それらの詳細については別稿[7,8]で報告する.

2. 構成

このシミュレーション・システムは，アドレスパターンの発生部（以下アドレス部と略）と，キャッシュをシミュレーションする部分(以下キャッシュ部と略)の2つに分割され，いずれも，GPSS で書かれている．キャッシュ・メモリ・システムに対するアクセス要求は，必要なパラメータを持ったトランザクションをアドレス部がキャッシュ部に渡すことで行なわれ，それ以外の結びつきはないように，モジュール化してある．これは発生したアドレスパターンを容易にキャッシュ以外のシミュレーションに適用できるようにし，一般性を与えたためである.

3. アドレスパターンの発生

3.1 プログラム構成

アドレスパターンは命令アドレス群とオペランドアドレス群の2種類から成ると考えられ，それぞれ強いローカル性を示すが，その個々の様態は若干異なる．また，リエントラントプログラミングの普及などによって両者の相関関係は非常に少くなってきた.

命令アドレスは，飛越しの場合を除き，逐次的に増加するが，この事自体強いローカル性を保障するものであるし，飛越しやオペランドアドレスのローカル性は，比較的せまい領域が間をおいて何度も現われるような性質であると考えられる．我々はワーキングセット[9]の概念を用いて，この種のローカル性を表わすことにした.

さて，アドレス部は，主ルーチン，ワーキングセット処理ルーチン（以下 WS ルーチンと略），および特性関数群により構成される．主ルーチンは，CPU の制御機構を簡略化したもので，キャッシュ部は，ここで作られたメモリー要求について，また WS ルーチンは個々のアドレス決定に際して，それぞれサブルーチンとして呼ばれる．特性関数群は，主ルーチンまたは WS ルーチンから必要に応じて呼び出される.

3.2 アドレス発生機構

アドレス発生機構における主ルーチンの役目は，CPU がメモリ要求を起す基本的状態制御を表現しようとするものである．命令ごとのメモリ参照は，命令

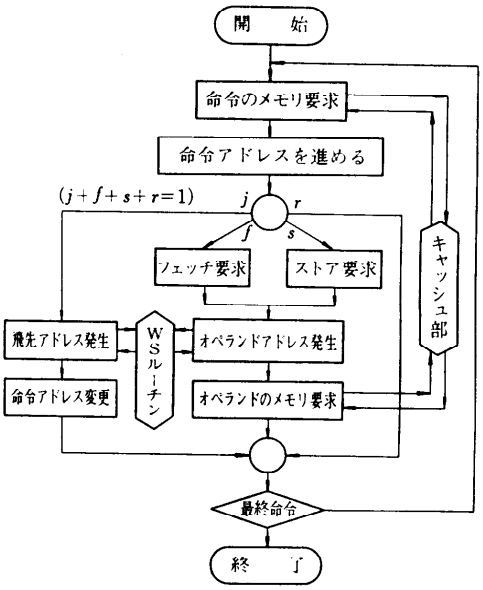

図 1 主ルーチンの流れ図
Fig. 1 Mainflow of address generation part

自体と，メモリ参照命令におけるオペランドについて行なわれる．前者は命令アドレスレジスタによって指示され，その内容は飛越しのとき飛先アドレスに書きかえられる以外は逐次的に増加する．これらのことはモデルにそのまま表現される．後者は，オペランドフェッチとストアの二通りあること，および逐次制御がなく命令アドレスレジスタに相当するものがないことの2点で前者と異なる．結局，主ルーチンの流れ図は図1のようになる．つぎに，WS ルーチンは，主ルーチンにおいてアドレス決定を必要とするごとに呼ばれるが，逐次的命令アドレスは主ルーチン自身が決定するので命令の飛先アドレスとオペランドアドレスの決定を受けもつ.

命令アドレスの決定方法は4種あり，確率 q_1, q_2, q_3, q_4 で発生する．q_1 は逐次的，q_2 は現在の命令が属する仮想ページの内部への飛越し，q_3 はワーキングセットに属するページの1つを選んで，その中へ飛越し，q_4 は主メモリ全域へのランダムな飛越しを意味する．ここでワーキングセットは $w(t,\tau,p)$ と表わされ，時刻 t においてそれより過去 τ 命令実行の間に参照されたページの集合であって，p はページの大きさであり，現在の命令が入っているページは $w(t,\tau,p)$ に含めない，q_2 は条件テストや小ループ等による飛越しを表わし，q_3 はサブルーチン間のやりとり等による飛距離そのものは大きいが，過去に使われたアドレ

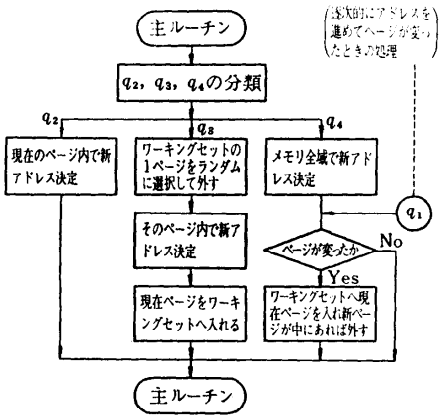

図 2 WS ルーチンの流れ図
Fig. 2 Flow of working-set control routine

スの近辺へ度々戻ることを意味し，アドレスのローカル性を表現している．q_4 は全く新しいプログラムを開始する意味であるから，その確率は小さく，飛越しの大半は q_2+q_3 によって与えられる．

一方，オペランドアドレスの決定においては，q_1 は非常に小さく，q_2, q_3 が大きいとみられる．これ以外の点は，命令アドレスの場合とほぼ同様である．

ワーキングセットを GPSS で実現するにあたって，我々はユーザチェインをワーキングセットの容器とし，要素となるページ情報を，前述のようにトランザクションとしてこの中に入れることとした．図2に WS ルーチンの流れ図を示した．ワーキングセットは命令，オペランド別々に用意したが，ルーチンを構成するブロック群はすべて共用である．

3.3 プログラム動特性の表現

アドレス発生の立場から，プログラムの動特性を，(1)命令頻度，(2)命令長，(3)飛先アドレス，(4)オペランド長，(5)オペランドアドレスのように整理する．このうち(3)，(5)はすでに述べた．

(1)の命令頻度は命令セットにおける各命令の発生確率であるが，必要なものはつぎの4種に分類される．

J：飛越命令，F：オペランドフエッチ命令，
S：オペランドストア命令，R：その他の命令．

J は飛越成立のみとし，R はレジスタ参照命令や，一般の非メモリ参照命令から成るが，不成立の飛越命令もこれに含める．前述の q_1 は F, S, R の発生頻度の和であり，$q_2+q_3+q_4$ が

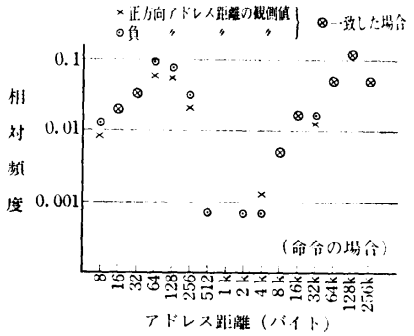

図 3 アドレス距離の分布（命令の場合）
Fig. 3 Freequenicies of address distance

J の頻度に当り，(2),(4)は可変語長方式を想定して設けた．以上の設定値は IBM 360 のアーキテクチャを背景とし，Hitac 8400 の実測値などから決定した．

特性関数群はこれらを GPSS の関数で定義したものであって，必要に応じて内蔵の一様乱数を用いる．この方法によれば，動特性の設定値を変更しながら反復シミュレーションする場合，特性関数群の定義カードを変更するだけで済ますことができ，便利である．

3.4 実験結果

アドレス部のシミュレーションについて，観測結果を代表的なもののみ挙げる．図3はアドレス距離（前回と今回の発生アドレスの差）の分布の例で，図4はワーキングセットの微視的推移の一例である．シミュレーション開始より 200 命令について，ページ換えごとに表示したものである．平均ワーキングセットサイズは，10^4 命令試行して命令の場合約3，オペランドの場合 13 前後であった．

4. キャッシュ・メモリのシミュレーション

4.1 プログラム構成

キャッシュ部は図5に示したようにキャッシュ・メモリのイメージとそれを制御するプログラムから成る．図において，“サーチ”は要求されたアドレスのデータがキャッシュ中にあるかどうかを調べるルーチ

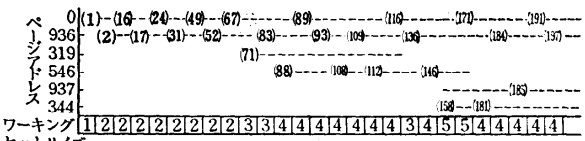

図 4 ワーキングセットの微視的推移
Fig. 4 Contents of working-set

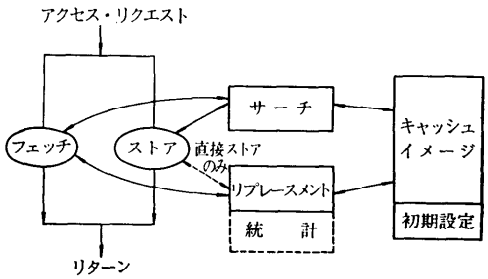

図 5 キャッシュ・シミュレーション部の構成
Fig. 5 Program structure of the cash simlation part

ン，また“リプレースメント”ルーチンは，そのデータがキャッシュ中にない時に，入れ換えアルゴリズムにしたがってキャッシュ中のブロックを追出し，新しいブロックを割り当てるためのものであって，いずれも実際の計算機内でハードウェアが行なっている操作を，そのままプログラムしたもので，パラメータの設定が簡単に行なえるよう考慮した以外は標準的なものである．

4.2 キャッシュのイメージの実現法

GPSS でキャッシュのイメージを作る方法は，次の3通り考えられる．

① マトリックスセーブバリュによるもの，
② 普通のセーブバリュの組合せ，
③ ユーザチェインの利用．

どの場合も，データの値そのものを記憶する必要はないが，マップされた主記憶アドレス，データの有効性，入れ換えアルゴリズムが利用するアクセス順序情報，割り当て時間，アクセス回数等の統計情報を記憶しておく必要があり，“サーチ”および“リプレースメント”ルーチンが高速になる形式であることが望ましい．

キャッシュメモリは直接形マッピング以外では，2次元（場合によっては3次元）の構造を持っているので，①の方法が最も直観的でわかり易いが，“サーチ”ルーチンは GPSS のブロックのレベルで逐一比較する方法を取らざるを得ないため時間がかかる．また我我が主として用いた Hitac-8400 の GPSS にはこの機能がなく，①をプログラムでシミュレートする②の方法を用いざるを得ないので，その場合は，更に低速になる欠点がある．一方，③はユーザチェインに，トランザクションを一定の規則で順番に並べておく機能や，チェイン中のトランザクションのパラメータ値でサーチを行なわせる機能があるので，内部のトランザクションをキャッシュのブロックに対応させると“サーチ”，“リプレースメント”サブルーチンが高速化され，プログラムも簡単になって，ユーザチェインの使い方としては，やや変則的だが都合がよい．したがって，直接形マッピングではアソシアティブサーチがなく，この方法はかえって繁雑なので②を利用したが，それ以外は全て③でイメージを実現した．

4.3 シミュレーション・パラメータの設定

キャッシュメモリシステムのパラメータは，命令セット，バス幅等のアーキテクチャのような間接的なものから，容量，マッピング方式等の直接的なものまで極めて多数あり，それらの設定の仕方は結果に大きな影響を与えるが，我々の目的は，この新しい方法の妥当性の立証と，特に興味あるシミュレーションで今まであまり行なわれていない場合についての結果を求めることであったので，基本的にはキャッシュメモリを初めて実用化した IBM 360/85 の諸パラメータを用いて，実験を行なった．その標準パラメータ値は，表1に示す通りである．またアドレスパターンに関するパラメータもいろいろ変えて，その影響を見るのは興味あることではあるが，今回は，マッピング方式とストア方式の影響とプログラムスイッチングの影響の実験にシミュレーションをしぼったので，やはり標準的パターンに対するパラメータを設定し，キャッシュのシミュレーションとしては，これを変えることはしてい

表 1 キャッシュメモリの標準パラメータ値
Table 1 Parameters of cache memory System

主記憶との速度比	13
容量	16 k バイト
ブロック長	64 バイト
主記憶・キャッシュ間バンド幅	1 ブロック＝64 バイト
セクタ内ブロック数	16
主記憶容量	512 k バイト

表 2 標準アドレスパターンのパラメータ
Table 2 Paramenters of the standard address pattern

命令長	20%（2バイト），74%（4バイト），6%（6バイト）平均 3.72 バイト
オペランド長	最大 256 バイト，平均 3.72 バイト（オペランドのあるものについて）
命令種類	ジャンプ：15%，ストア：10%，フェッチ：45% レジスタ(無オペランド)：30%
ジャンプ内分け	ページ内：40%，W-Set：50%，ランダム：10%
オペランドアドレス形	ページ内：50%，W-Set：40%，ランダム：10%
ワーキングセットのページの大きさ	命令：256，オペランド：512
ワーキングセットの L	50 命令（命令，オペランドとも）

表 3 標準パターンに対する結果（直接ストア，命令数 10^4）
Table 3 Results for the standard pattern (Immediate store, 10^4 instructions)

マッピング法		直接（コングリエント）	セットアソシアティブ(2)	セットアソシアティブ(4)	完全アソシアティブ	セクタ
実効 サイクル時間		3.462	3.376	3.323	3.322	3.255
実効 フェッチ時間		2.722	2.629	2.572	2.571	2.499
実効 ストア時間		13.000	13.000	13.000	13.000	13.000
命令 実行時間		7.873	7.677	7.558	7.555	7.403
命令 フェッチ時間		2.283	2.196	2.170	2.171	2.169
データ アクセス時間		10.136	9.935	9.768	9.761	9.453
平均ブロック（セクタ寿命）		5,930	6,554	68,38	6,987	1,842
平均ブロックアクセス回数		7.076	7.630	7.908	7.823	2.192
命令フェッチ確率 (%)	ブロック内 あり	95.0	95.5	95.7	95.6	95.5
	セクタのみ					3.4
	セクタ（ブロック）なし	5.0	4.5	4.3	4.4	1.1
データフェッチ確率 (%)	ブロック内 あり	66.2	67.5	68.7	68.7	70.8
	セクタのみ					22.1
	セクタ（ブロック）なし	33.8	32.5	31.3	31.3	7.1
データストア確率 (%)	ブロック内 あり	48.2	50.4	50.9	51.5	63.6
	セクタのみ					22.8
	ブロック なし	51.8	49.6	49.1	48.5	13.6

表 4 標準パターンに対する結果（スワップ，命令数 10^4）
Table 4 Routs for the standard pattern (swap, instructions)

マッピング法		直接（コングリエント）	セットアソシアティブ(2)	セットアソシアティブ(4)	完全アソシアティブ	セクタ
実効 サイクル時間		3.038	2.648	2.484	2.578	2.853
実効 フェッチ時間		2.854	2.495	2.435	2.431	2.693
実効 ストア時間		5.408	4.612	4.509	4.473	4.905
命令 実行時間		6.908	6.022	5.877	5.863	6.487
命令 フェッチ時間		2.447	2.200	2.169	2.169	2.375
データ アクセス時間		8.087	6.929	6.722	6.698	7.455
平均ブロック（セクタ）寿命		4,791	4,724	4,909	4,993	1,267
平均ブロックアクセス回数		6.674	7.248	7.564	7.494	7.737
命令フェッチ確率 (%)	ブロック内	94.9	95.5	95.7	95.7	94.5
	セクタのみ					4.2
	ブロックなし	5.1	4.5	4.3	4.3	1.4
データフェッチ確率 (%)	ブロック内	69.5	71.0	72.2	72.3	68.1
	セクタのみ					24.7
	ブロックなし	30.5	29.0	27.8	27.7	7.3
データストア確率 (%)	ブロック内	68.6	69.9	70.8	71.1	67.5
	セクタのみ					25.2
	ブロックなし	31.4	30.1	29.2	23.9	7.3

ない．そのパラメータ値は表2に示す通りである．

4.4 マッピング方式とストア方式の比較

まず，第1の実験では上記標準パラメータ条件のもとで，5種類のマッピング方式に対し，ストアの処理を直接ストアにした場合とスワップ形にした場合について，各種データを測定したところ，直接ストアについては表3に，スワップの場合については表4に示したようなデータが得られた．シミュレーションした命令数はいずれも 10^4 命令であって，一見，少ないようであるが，2,000 命令ずつ中間結果をとって，比較したところマッピング方式によって若干の違いはあるが，この数の命令までに完全に安定した結果が得られている．これは発生されたアドレスパターンのばらつきが小さいためと考えられる．

このシミュレーションデータから，およそ次のようなことを読みとることができる．

まず直接ストアの場合について，

（1） 直接マッピングから完全アソシアティブへ向って，マッピングブロック数†が増加するにつれ，実効サイクル時間の改善が認められるが，差はあまり大きくなく，セットアソシアティブ(4)†と完全アソシアティブではほとんど有意差はない．したがって，マッピングブロック数を4以上にすることはサーチ時間，ハードウェア量の増大を考慮すれば無意味で，むしろ損失を招くことがわかる．

（2） 実効サイクル時間は 3.3（主記憶とキャッシュ比の約25%）前後であるが，キャッシュの効果のないストアを除いたフェッチ時間では約 2.5（約20%），直接マッピングですら 2.7（21%）とキャッシュの有効性が立証されている．

（4） セクタ形が最も結果がよい．これは標準パターンのオペランドのアドレスパターンのばらつきが比較的小さいためであろう．

スワップ形ストアの場合は，

（4） 直接形ストアより実効サイクル時間は 10～20% も改善される．

（5） マッピング方式による差がより強く表れ，直接形で最もよかったセクタ形はセットアソシアティブ(2)よりも悪くなる．これはセクタ単位の割当てのためにむだなスワップが多くなるためと考えられる．

† 主記憶内の各ブロックがマッピングされ得るキャッシュ内のブロック数．

† （ ）内はマッピングブロック数（セット内ブロック数に同じ）．

その他の結果は，

（6） 平均ブロック寿命は直接ストアの場合 6,000～7,000 サイクル，スワップの場合 4,700～5,000 サイクルで，命令数になおすと，それぞれ 750～930，700～800 命令に相当する．セクタ寿命はこれよりずっと小さい．表には平均値しか示されていないが，いずれの場合も，ばらつきが極めて大きいのが特徴である．

（7） ブロック平均アクセス回数はいずれも 10 以下で小さいが，これもばらつきが非常に大きく，150 回ぐらいアクセスされたブロックもある．アドレスパターンにローカル性があるのだから，当然の結果といえる．

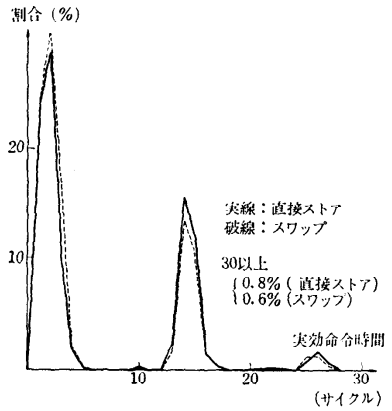

図 6　実効命令処理時間の分布(セクタ形)
Fig. 6　Distribution of effection execution time (sector mapping)

（8） セクタ形の場合のセクタ内ブロック利用率は，直接ストアで 29%，スワップで 27% とあまりよくない．

（9） 命令の実効処理時間分布の例として，セクタ形マッピングの場合を図6に示した．この図の3つのピークは左から主記憶のアクセスが，0, 1, 2回にそれぞれ対応し，スワップ形ストアでは最左のピークがより大きくなって，性能向上が現われている．

4.5 プログラムスイッチの影響

第2の実験では，このアドレスパターン発生方式の融通性を利用し，マルチプログラミング時のようなプログラムスイッチングが行なわれる情況を模擬し，その時のキャッシュ・メモリの効率への影響をシミュレーションしてみた．

4.5.1 方法

アドレスパターンのパラメータは，前と同じものを用いるが，ワーキングセットを2個ずつ用意し，それを一定の命令数（スイッチ周期）ごとに切り換えて，アドレス発生を行ない，2個のプログラムがこのスイッチ周期ごとに交互に走る情況を模擬したパターンの発生を行なって，キャッシュ・メモリの動作をシミュレーションした．もちろん，ワーキングセット数を増し，3個以上のプログラムや，それぞれの走行周期が異なるような状況をシミュレーションするのも容易であるが，我々はスイッチ周期の変化と，マッピング方

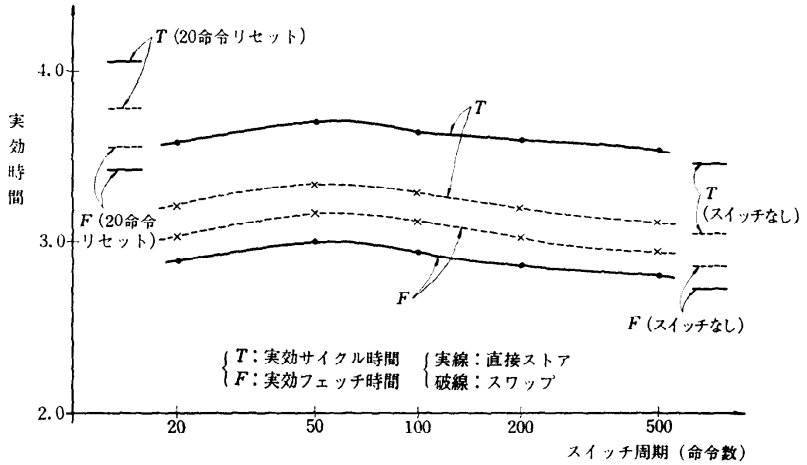

図 7　プログラムスイッチの影響（直接形マッピング）
Fig. 7　The influence of program switching (direct mapping)

式，およびストア方式によるその影響の出方の違いを調べることを中心に考え，2つのプログラムが対称に同じ周期で，切り代る場合にしぼった．更に，極端な状況としてワーキングセットを最小のスイッチ周期ごとに完全にリセットしてしまう場合についても同時に測定した．これは小さな互いに無関係なプログラムが，次々と走る場合を表わしているから，キャッシュ・メモリ方式にとっては最も厳しい状況であり，その効率の悪い方の限界をほぼ与えるものと思う．マッピング方式は，第1の実験で差が小さいことがわかったものを省き，直接，セクタ，セットアソシアティブ(4)の3つだけについて，それぞれ直接ストアとスワップの場合について，シミュレーションした．またスイッチ周期は対数的に 20, 50, 100, 200, 500 の5種の場合を取りあげた．図7～9は最も重要な実効サイクル時間と実効フェッチ時間について，スイッチ周期との関係をグラフに示したものであるが，このシミュレーションの結果から，ほぼ次のようなことがわかった．

(1) いずれの場合もスイッチ周期が 50 の時が一番性能が悪く，20 になると再びよくなる．これは 20

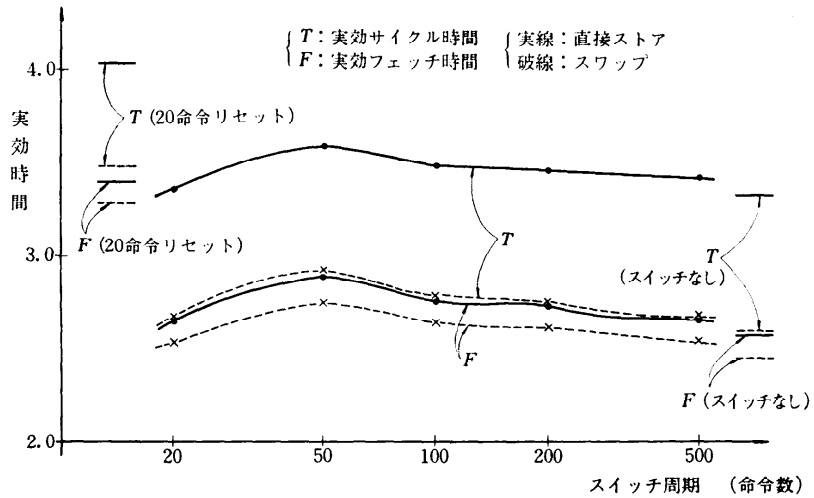

図 8 プログラム・スイッチの影響（セクタマッピング）
Fig. 8 The influence of programm switching (sector mapping)

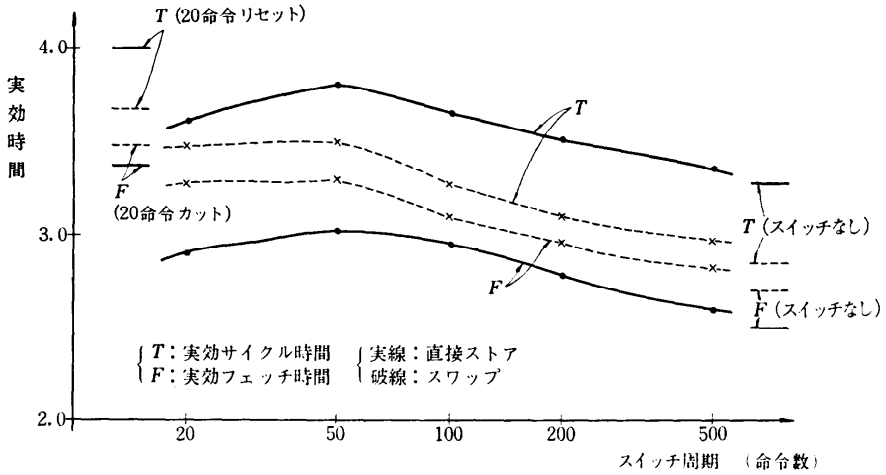

図 9 プログラム・スイッチの影響（セットアソシアティブ(4)）
Fig. 9 The influence of program switching (set associative (4))

命令の休みではまだ前の情報が残っているためだろう.

(2) スイッチ周期によって,実効サイクル時間が最も大きく変化するのはセクタ形で,アソシアティブ性を持たない直接形はほとんど影響を受けない.100命令以下ではセクタ形の性能は直接形と変らず,セクタ形はマルチプログラミング情況に向かない方式といえる.セットアソシアティブ形は直接形同様,スイッチ周期の影響を受けにくいのに加え,性能はずっとよく,ユニプログラムでもセクタ形とほとんど変らないから,この実験結果から判断する限り,最もよいマッピング方式であるといえる.20命令リセットの悪条件でも,実効サイクル時間は記憶速度比の30%強(直接ストア),または26%前後(スワップ)が得られる.

(4) プログラムスイッチの影響度のストア方式による変化はほとんどない.

5. むすび

以上,乱数をもとにした新しいプログラムアドレスパターンの発生方式および,それをキャッシュメモリのシミュレーションに応用して得られた種々の興味ある結果について報告した.

このアドレスパターン発生方式は融通度が高く,実際のプログラムの場合に類似したパターンを容易に発生でき,かつ特殊ケースの模擬も簡単であり,プログラムアドレスパターンの関係するシステムのシミュレーションに一般的に有効であろう.ただ残念なことに,実際のプログラムのアドレスパターンがあまり手もとになかったため,その正しさやパラメータ値と実際のプログラムの関係等を定量的に検討することはできなかった.

一方,キャッシュ・メモリのシミュレーションにおいては,各種のアドレスパターンについて実験しなかったので,これをそのまま一般的結論とするのは危険であるが,これまであまり知られていなかったキャッシュメモリの動作データが得られ,プログラムスイッチングの影響の実験を行なったことで,セットアソシアティブ形マッピングがセクタ形よりすぐれていることを,ほぼ立証できたようにわれる.その他のデータも一応の目安としては用いることができよう.

もちろん,不備な点はまだ残っているが,ここで提案するアドレスパターン発生方式がかなり有効に使えそうであること,および,キャッシュ・メモリのシミュレーションでも2,3の新しい結果が得られ,必要なら個々のシステムにこの方法を応用し,より詳しいデータを得ることができる見通しがついたことで,我々の初期の目的は一応達成することができたと思う.

終りに,この研究の機会を与えられた黒川一夫電子計算機部長,ならびに前部長野田克彦博士に謝意を表する.また御助言をいただいた慶応大学工学部相磯秀夫教授,および,討論,実験に協力をされた計算機方式研究室,可変構造計算機研究グループの諸氏に感謝します.

参 考 文 献

1) 飯塚 肇: キャッシュ・メモリ・システム(1~2), 情報処理, Vol. 13, No. 7, pp. 467~473 & No. 8, pp, 540~547 (1972).
2) 飯塚 肇: スレーブメモリを持つ計算機システムの簡単な解析, 電子通信学会論文誌 (C), Vol. 54, No. 8, pp. 698~705 (1971).
3) D. H. Gibson: Considerationin in block-oriented system design, Proc. SJCC, Vol. 30, pp. 75~80 (1967).
4) J. S. Liptay: Structual aspects of the System/360 model 85. II. The cache, IBM Systems J. Vol. 7, No. 1, pp. 15~21 (1968).
5) 加納 弘: バッファメモリ方式のシミュレーション, 電気学会論理装置の設計自動化専門委員会資料 (1969).
6) 飯塚 肇, 照井武彦: GPSS によるアドレス発生とキャッシュ・メモリのシミュレーション実験, システム評価シンポジウム報告集, pp. 162~175 (1972).
7) 飯塚 肇, 照井武彦: プログラムアドレスパターンの発生について, 電子総研彙報, 投稿予定.
8) 飯塚 肇: キャッシュメモリのシミュレーション, 電子総研彙報, 投稿予定.
9) P. J. Denning: The working set model for program behavior, CACM, Vol. 11, No. 5, pp. 323~333 (1968).

(昭和48年5月8日受付)