解　説

キャッシュ記憶†

長島重夫††　堀越　彌††

1. はじめに

キャッシュ (Cache) 記憶は，またバッファ記憶，ローカル記憶などとも呼ばれ，CPU 内にある高速アクセスが可能なメモリである．一般に CPU の内部処理速度に比べて主記憶装置 (Main Storage, MS) へのアクセス速度は数～十数倍も遅く，処理に必要な命令，データを命令の実行ごとに読出していたのでは，命令の高速処理は望めない．そこで MS の内容の一部を CPU 内部に置く高速メモリにコピーし，大部分のメモリ参照をその高速メモリに対して行うようにして処理の高速化をはかる方式が考えられた．この高速メモリがキャッシュ記憶（以下では単にキャッシュと呼ぶ）である（図-1 参照）．通常，キャッシュへのアクセス時間は CPU のマシンサイクル時間と同程度になっている．

キャッシュが商用機に取入れられたのは 1968 年に IBM が発表した 360/85[11] が初めてである．その後，IBM は 360/195, 370/155[14], 165[16], 195, 158[15], 168[17], 303 X シリーズ[18] のように大型機のすべてに採用し，日本でも大型プロジェクトの超高性能電子計算機[25]，電電公社の DIPS-1[26),27]，DIPS-11 シリーズをはじめ，各社の大型機に採用されている[24]．キャッシュの容量は当初は 8 KB 程度であったが，現在の大型機は 64 KB が標準的な容量となっている[12]．表-1 に現在の代表的計算機のキャッシュの構成を示す．

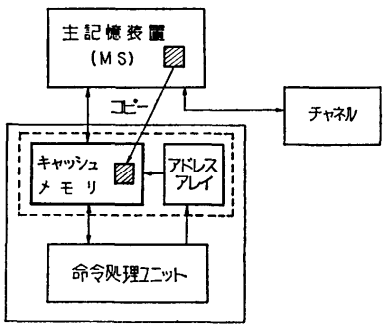

図-1　キャッシュ記憶の位置付け

表-1　大型機のキャッシュの構成例[12),18)~23),48)]

項　目	IBM 3033	日立 M-200 H	富士通 M-200	日電 ACOS-900
キャッシュ容量	64 KB	64 KB	64 KB	8 KB
キャッシュ素子		1 Kb	1 Kb	1 Kb
ブロック容量	64 B	64 B	32 B	16 B
マッピング方式	セットアソシアティブ	セットアソシアティブ	セットアソシアティブ	セットアソシアティブ
構　成	16ロー×64カラム	16ロー×64カラム	16ロー×128カラム	4ロー×128カラム
リプレースメントアルゴリズム	modified* LRU	4グループ分割 LRU	LRU	FIFO

* 具体的方式は不明

2. キャッシュの目的と効果

キャッシュの目的はいうまでもなく CPU の高速化である．前述のようにキャッシュは MS のコピーであり，大部分のメモリアクセスはキャッシュにて処理し，必要なデータ（命令も含む）がキャッシュにないときのみ MS から必要なデータを含む数 十 B（これをブロックと呼ぶ）を転送する（コピーする）．これをブロック転送と呼び，これにより将来必要とするデータをあわせて転送しておくことがキャッシュの効果を生み出している．すなわち，短時間でみれば，プログラムの参照するメモリ容量は MS 容量に比べて極めて小さい，というプログラムの特性をうまく利用しているといえる．

キャッシュを有する計算機の性能は，概略，次式で表わすことができる．

$$\bar{T}=T_0+\beta\cdot T_B$$

ここで，T_0 はすべての命令，データがキャッシュ上にある場合の CPU の平均命令実行時間，β はデータがキャッシュ上にない命令当りの確率，T_B はブロック転送時間，$\bar{T}$ は CPU の平均命令実行時間である．

† Cache Memory by Shigeo NAGASHIMA and Hisashi HORIKOSHI (Central Research Laboratory, Hitachi Ltd. 8th Department).

†† (株)日立製作所中央研究所

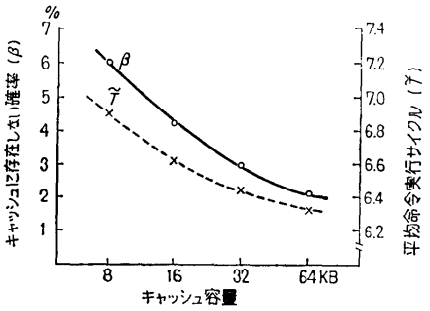

図-2　技術計算プログラムにおける
キャッシュ効果の一例

したがって，$\tilde{T}$ を小さくする（高速化する）ためには，キャッシュの観点からみれば，β および T_B を小さくする必要がある．β を小さくするためには，

（1）　キャッシュの容量を増加する．

（2）　ブロックの容量を大きくする．

（3）　MS とキャッシュ間のブロックの対応（マッピングと呼ぶ）手法を工夫する．

（4）　リプレースメント・アルゴリズム（後述）を工夫する．

が重要である．また，T_B を小さくするためには，

（1）　MS のアクセス時間を小さくする．

（2）　ブロックの容量を小さくする．

（3）　論理的に T_B を小さくするよう見せる．

などが重要である．一般にキャッシュはキャッシュにない確率 β のみが注目されがちであるが，ブロック転送時間 T_B を小さくすることも同等に重要である．このほか，キャッシュは CPU 内に存在することから，チャネルや他 CPU からの MS へのストアによる MS とキャッシュの不一致の除去の方法も重要である．これらについては次章に詳細を述べる．

図-2 は技術計算プログラムにおけるキャッシュの効果の一例をキャッシュ容量をパラメタとして示したものである．同一容量でもプログラムの特性により β は 1～20% と大幅に異なり，一般に科学技術計算では小さく，オンライン等のシステム的な処理では大きい傾向がある．

3. キャッシュの制御方式

3.1 MS とのマッピング方式

前述のように，MS とキャッシュはブロックを単位としてマッピングされる．メモリアクセス時には，必要とするデータがキャッシュ上にあるか，またどこにあるかを調べるために，アクセスするアドレスとキャッシュ上に存在する全ブロックのアドレスを格納したアドレスアレイとを比較する．一致するアドレスがあれば，それに対応したデータをキャッシュから読出し，なければブロック転送すると同時に，アドレスアレイにもそのデータのアドレスを登録しておく．このアドレス比較はキャッシュの高速アクセスのために短時間で行う必要があるが，アドレスアレイには容量（標準で 2 KB 程度必要）の点からメモリを使用し，かつ，アドレス比較器の数を少なくしたいという要求がある．これらの要求を満たすために，次に述べるような各種のマッピング方式が提案され，実施されている．

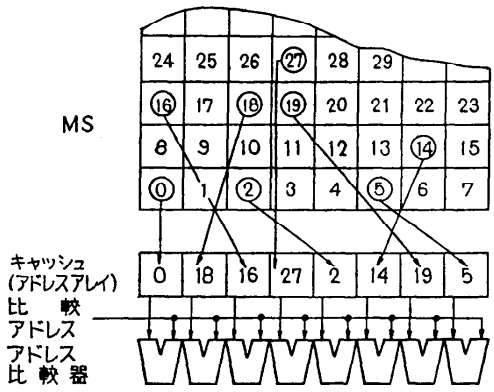

図-3　アソシアティブ方式マッピング

（1）　アソシアティブ方式（**図-3** 参照）

この方式は MS 上のブロックをキャッシュ上の任意の位置にマッピングしてもよい方式である．したがって，キャッシュをアクセスするには全アドレスアレイ中のアドレスと同時に比較する必要があるが，キャッシュのブロック数は 256～1,024 にも及ぶことからアドレス比較器も同数必要となり，現実的でない．このため，本方式を採用した計算機は今の所は存在しない．しかし，キャッシュの使用効率は最もよい．

（2）　セクタ方式（**図-4** 参照）

この方式は MS 上の連続したブロックをまとめて 1 セクタとし，セクタ単位でアソシアティブ方式を採用してアドレス比較器の数を減少させる方式である．セクタ内は図-4 で示すように MS と 1：1 の対応でマッピングする．通常，セクタの容量としては 1～2 KB が採用され，したがってアドレス比較器は 8～16 個ですむ．本方式は IBM 360/85[11] で採用された方式であるが，キャッシュの使用効率が上らず β が小さくならない傾向がある．

（3）　コングルエント方式（**図-5** 参照）

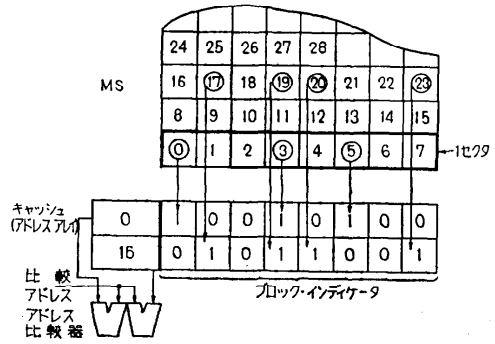

図-4　セクタ方式マッピング

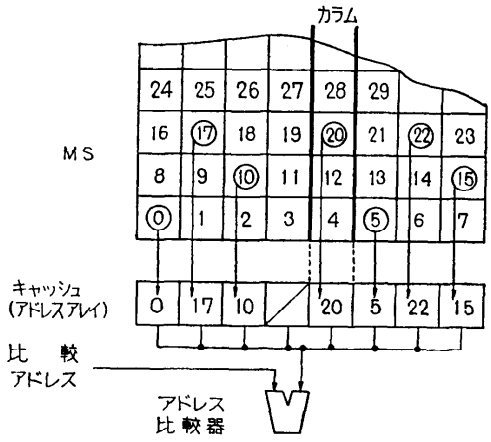

図-5　コングルエント方式マッピング

この方式はMSおよびキャッシュ上のブロックをいくつかのカラムに分割し，マッピングは同一カラム内に固定する方式である．MS 上の同一カラム内の二つのブロックがキャッシュ上に存在することはできない．本方式の最大の特徴は，カラムの番号をアクセスするアドレスから一義的に決定できることにより，わずか1個のアドレス比較器ですませることができることにある．しかし，マッピングが固定されていることからキャッシュの使用効率が低い．

（4）　セットアソシアティブ方式（図-6 参照）

この方式は(3)のコングルエント方式に対し，キャッシュ内の各カラム（64～128 カラムが一般的）中のブロック数（これをロー数と呼ぶ）を増加させた方式である．この方式はカラム数を1とすると(1)の方式に，ロー数を1とすると(3)の方式とみることができる．本方式は，アドレスアレイにメモリを使用できること，アドレス比較器がロー数だけですむこと，キャッシュの使用効率がよいことから，IBM 370/165 で採用されて以来，大部分の計算機に採用されている[24]．以後は本方式を中心に説明する．図-7 に同一のキャッシュ容量における各方式による β の違いの一例を示す．

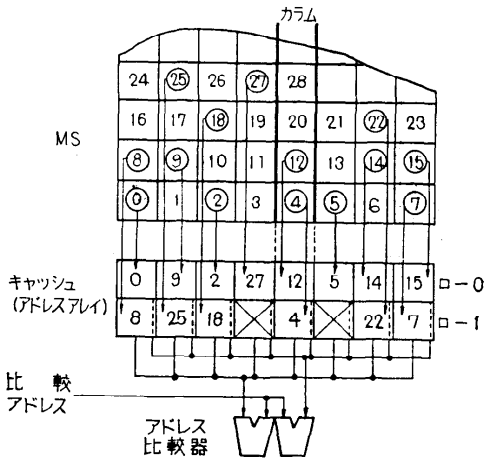

図-6　セットアソシアティブ方式マッピング

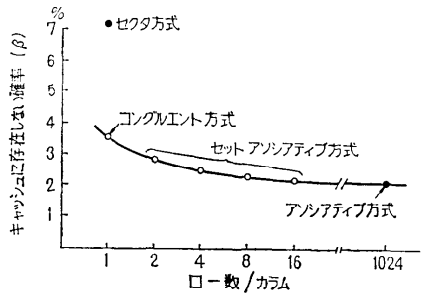

図-7　同一キャッシュ容量における各方位による β の差の一例

3.2　リプレースメント・アルゴリズム

ブロック転送時にキャッシュ上のあるブロックを選択して新しいブロックと置換するが，そのブロックの選択アルゴリズムをリプレースメント・アルゴリズムと呼ぶ．それには，次の各方式がある．

（1）　LRU (Least Recently Used) 方式

この方式は最も以前に参照されたブロックを置換する方式である．セットアソシアティブ方式では各カラム内でこの制御を行う．この方式は，最も以前に参照されたブロックは，以後最も使用されないであろうという仮定によっているが，現状の各種の使用環境において安定した良さ（低い β 値）を示すといわれている[33]．LRU 制御を実現するためには，キャッシュへ

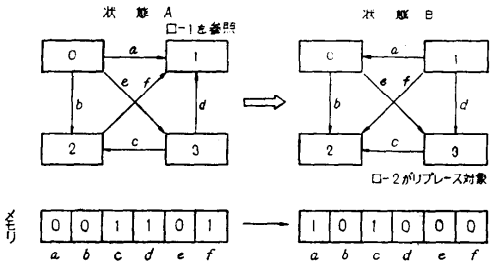

図-8 LRU リプレースメントアルゴリズムの実現方式の一例[21]

のアクセスごとにブロックの参照履歴を記録しておく必要がある．このためにセット・アソシアティブ方式では，4ローの場合，各カラムごとに6ビットのメモリを設け，次のように制御する（図-8参照）[21]．すなわち，各ローのブロックの参照の順位を6個の矢印で表わし，各ブロックに向けられた矢印の向きの数が多いほど，過去に参照されたと定義しておく．図-8の状態Aでは，ロー1のブロックが最も以前に参照されたブロックである．この矢印の向きをメモリの0/1に対応させて記憶する．あるブロック（状態Aではロー1のブロック）を参照すると，そのブロックに向けられた矢印の向きをすべて反転させる（実際にはメモリの内容を更新する）．リプレースすべきブロックを決定するときには，すべての矢印が向けられたブロック（状態Bではロー2）を選択する．この決定は，メモリを読出してデコードすればよい．この方式では，4ローのときにはメモリは6ビットでよいが，8ローでは28ビットも必要になる（${}_8C_2=28$）．このビット数を減少させるために，ローをいくつかのグループに分割し，グループ間およびグループ内で各LRU制御を行う方式が採用されることもある[20]．

（2） 疑似 LRU 方式

各ブロックに1ビットの参照ビットを置き，初期値を0とする．あるブロックをアクセスするとそのブロックの参照ビットを1とする．すべての参照ビットが1になると最後にアクセスしたブロックの参照ビットを除き，すべて0にする．リプレースは参照ビットが0で最も番号の小さなブロックを対象とするという方式で，アドレス変換用のアドレスアレイの制御としてIBM 360/67 に採用されたが，セットアソシアティブ方式のキャッシュ制御には採用されていない．

（3） そ の 他

その他，FIFO (First In First Out，最も過去に登録されたブロックを選択する)[24] やランダム（ランダムに選択する）方式がある．

3.3 ブロック転送方式

2章に述べたように，ブロック転送時間 T_B の短縮も重要である．β を小さくするにはある程度までブロックの容量を大きくすることが有効であるが，これは T_B の増大につながる．現在の大型機では，64 KB キャッシュでは 32 または 64 B ブロック，32 KB 以下のキャッシュでは 32 B ブロックが一般的である．ブロックの容量は，また，MS とのインタリーブ数にも関係する．大型機では通常8B単位でMSとデータ転送するが，32 B ブロックの場合，ブロック転送時には四つのメモリバンクを同時に（または1サイクルごとに）起動し（4ウェイインタリーブ），8Bのデータを1サイクルごとに受取ってキャッシュに書込む（図-9参照）．このように，できるだけ MS を並列に読出すようにして T_B の短縮をはかっている．64Bブロックの場合は8ウェイ以上のインタリーブが必要である．このほか，T_B の短縮のために，次のような技術が採用されている．

（1） ブロック転送の順序制御

ブロック内の必要なデータが最初にキャッシュに到着するように制御し，残りのデータのキャッシュへの書込みと命令の実行をオーバラップさせる方式である．

（2） ブロック転送データのバイパス制御

ブロック転送時，必要なデータをキャッシュに書込むと同時にバイパスして命令処理ユニットに送ることはもちろん，ブロック転送の残りのデータすべてを命令処理ユニットに送ることにより，1命令で多数のデータを必要とする場合でもキャッシュからのデータ読出しを不要とする方式であり，IBM 3033 にて採用さ

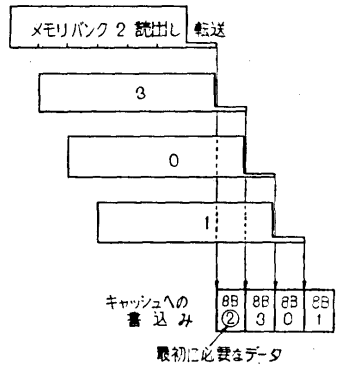

図-9 ブロック転送時の MS の起動

れている[13].

3.4 キャッシュへのアクセス方式

キャッシュをアクセスするにはアドレスアレイを見てキャッシュ内のブロック位置を知る必要がある．また，現在の計算機は仮想アドレスをサポートしているため，キャッシュをアクセスするにはアドレス変換も加わることになる．したがって，アドレス変換⟶アドレスアレイ索引⟶キャッシュのアクセスの時間の短縮が重要となる．アドレス変換はアドレス変換対が登録されている高速メモリ（これを TLB——Translation Lookaside Buffer と呼ぶ）をアクセスすることにより行われる．TLB，アドレスアレイ，キャッシュのアクセスの順序により，次のような制御方式に分けられる．

（1） シリアル方式

TLB，アドレスアレイ，キャッシュを順次アクセスする方式である．もっとも時間を要するので，ほとんど採用されていない．

（2） シリアル——パラレル方式

TLB をアクセス後，アドレスアレイとキャッシュを同時にアクセスする方式である．アドレスアレイと同様，キャッシュも指定されたカラム中のロー数だけのブロックを同時に読出し，一致したアドレスアレイに対応するブロックのみを選択するわけである．

（3） パラレル——シリアル方式

TLB とアドレスアレイを同時にアクセスしてキャッシュ内のブロック位置を決定した後，キャッシュをアクセスする方式である．

（4） パラレル方式

TLB，アドレスアレイ，キャッシュすべてを同時にアクセスする方式である．（2），（3）を結合した方式と考えることができる．

以上の方式の中で，大型機では（1）が採用されることはなく，大部分が（2）または（3）の方式を採用している．中には（4）を採用している計算機もある．いずれの方式を採用するかは，アクセスに必要なマシンサイクル数とマシンサイクル時間の積，および必要ハードウェア量を勘案して決定するが，全般としては，必要マシンサイクル数は（1）＞（2）≈（3）＞（4）である反面，マシンサイクル時間は（1）＜（2）≈（3）＜（4）の関係にあり，また，必要ハードウェア量は（1）＜（3）＜（2）＜（4）となる．

3.5 ストア方式

キャッシュへのストアの方式は次の二方式がある．

（1） ストア・スルー方式

ストアすべきアドレスがキャッシュ内に存在するときには，キャッシュと MS の双方にストアする．常にキャッシュと MS を一致させる方式である．

（2） ストア・スワップ方式

ストアすべきアドレスがキャッシュ内に登録されている場合にはキャッシュのみにストアし，MS にはストアしない方式である．キャッシュ内に登録されていない場合の処理は，さらに二方式に分けられる．

（a） ストアすべきアドレスに対応するブロックをキャッシュに転送し，そのブロックにストアする方式

（b） 上記ブロック転送せず，MS に直接ストアする方式

いずれにせよ，ストア・スワップ方式では，更新されているブロックがリプレースの対象に選択されるとそのブロックを MS にストアしてからブロック転送を行う．

（1），（2）いずれの方式を採用するかは，いろいろの観点から判断しなければならないが，おおよそ次のようにいうことができよう．

（1） ストア命令を短時間で完了させるためには，（2）がすぐれている（MS へのストアが不要のため）が，（1）の方式でもストアのおいてきぼり制御の併用により，同等の性能を実現することは可能である．

（2） ブロック転送時間は（2）の方が長い（ストアされているブロックを MS にストアするため）．

（3） ブロック転送の確率 β については，（2）（a）の方式は（1）より多い（（2）（b）方式と（1）はほぼ同等）．

（4） 同一ブロックに何回もストアできるため，MS の動作時間（バンクビジー時間）は（2）の方が少ない．

ストアの方式は，このほか，マルチプロセッサやチャネルの構成とも深い関係がある．それについては次節に述べる．IBM は 360/85 以来 303 X シリーズまですべてストア・スルー方式を採用してきたが，4341 プロセッサで初めてストア・スワップ方式を採用している[19]．

3.6 メモリシェア時における MS・キャッシュ一致方式

キャッシュを制御する上での大きな問題点としてメモリシェア時のキャッシュの一致方式である．これは，

（1） マルチプロセッサ構成における各 CPU 内の

キャッシュの一致方式

（2） チャネルからのアクセス時のMSとキャッシュの一致方式

の二つの問題に分けられる．マルチプロセッサ構成をとらない計算機では(2)の問題のみを考えればよい．

（1） マルチプロセッサ構成における各CPU内のキャッシュ一致方式

（a） ストア・スルー方式の場合

ストア・スルー方式の場合の制御は比較的容易である．すなわち，ストア時にストア・アドレスを他CPUに送り，受取ったCPUは自キャッシュ内にそのアドレスに対応するブロックがあるか調べ，あればそのブロックを無効にする．無効にされたブロックをアクセスすると再びブロック転送を行うが，ストア・スルー方式ではMSのデータはすでに更新されているから，新しいデータを含むブロックが転送されることになる．

（b） ストア・スワップ方式の場合[28]

ストア・スワップ方式の場合は，若干制御が複雑である．一般にMS上のあるブロックは複数のCPU内のキャッシュにコピーされているが，ストアによるブロック更新時には，1台のCPU内のキャッシュのみを更新し，他CPUのブロックは無効とする（更新されているブロックとされていないブロックが各CPUに同時に存在することはない）ように制御する．そのため，ブロック転送時およびストア時には，他CPUにそのアドレスを送出する必要がある．制御の詳細を**表-2**に示す．このアドレス交換の頻度を減少させる方式も提案されている[29]．

（2） チャネルからのアクセス時のMSとキャッシュの一致方式

チャネルからのメモリ・アクセスはCPU内のキャッシュとは無関係にMSと直接行われることが多い．この場合にも，チャネルからのストア時には(1)と同じ状況が生じ得るため，チャネルのストア・アドレスとキャッシュのアドレスアレイを比較し，一致するアドレスがあれば，それに対応するブロックをキャンセルする．ただし，IBM 370/165では，チャネルからのストア・データをキャッシュにもストアしている[16]．また4341ではチャネルへ送るデータがキャッシュにあれば，キャッシュから読出している[19]．

いずれの場合にしても，他CPUまたはチャネルからアドレスを受取ったCPUは，自CPUの動作を停止させ，アドレスアレイを外部からのアドレスとの比較に使用するため，処理速度が低下する．この防止のために，同一内容のアドレスアレイを二面設け，一面は自CPUのアドレス比較に，一面は外部のアドレス比較に使用する方式も考えられる．

表-2 ストア・スワップの制御

アクセス ＼ ブロックの状態		アクセスするブロックが他CPUのキャッシュ上にあり，更新されている	アクセスするブロックが他CPUのキャッシュ上にあるが，更新されていない	アクセスするブロックが他のCPU上にない
フェッチ	ブロックがキャッシュ上になし	BS→BF→CF	BF→CF	BF→CF
	ブロックがキャッシュ上あり	——	CF	CF
ストア	ブロックがキャッシュ上になし	BS→BF→CS ——→CN	BF→CS CN	BF→CS
	ブロックがキャッシュ上にあり	——	CS CN	CS

BF ：ブロック転送
CF ：キャッシュからフェッチ
BS ：他CPUのキャッシュ上のブロックをMSにストア
CS ：キャッシュにストア
CN ：他CPUのキャッシュ上のブロックを無効化
——→：動作順序を示す
——：この組合せはあってはならない

4. 最近の大型計算機でのキャッシュ制御の一例

キャッシュを有する計算機の設計時には，以上に述べた中から，目標の性能を実現する構成，制御方式を選択することになる．しかし，実際にはさらに性能を向上させるために，あるいは実装上の制約から，各計算機ごとに制御上の工夫がなされることが多い．

そこで，最近の大型計算機におけるキャッシュ制御の一例として，HITAC M-200 Hを取り上げ，制御上の特徴について述べよう．M-200 Hのキャッシュ構成の概略については，表-1を参照されたい．

M-200 Hのキャッシュ制御は，1マシンサイクルに2回のアクセスを可能としたこと，アドレスアレイ用としてメモリとアドレス比較器を1チップに格納したLSIを開発したこと[30]が大きな特徴である．

（1） キャッシュの2アクセス/サイクルの制御

M-200 Hでは命令実行時間をできるだけ1サイクル/命令に近づけるために，高度のパイプライン制御を行っている．**図-10**は，ロード，加算命令のように，オペランドをメモリから読出してレジスタと演算する命令を次々と実行したときのパイプライン制御のようすを表わしたものである．ここでIF（命令読出し）

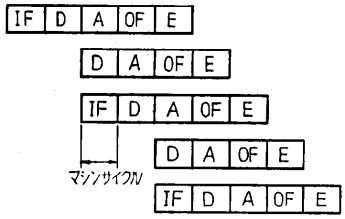

図-10　M-200H の命令実行ステージ

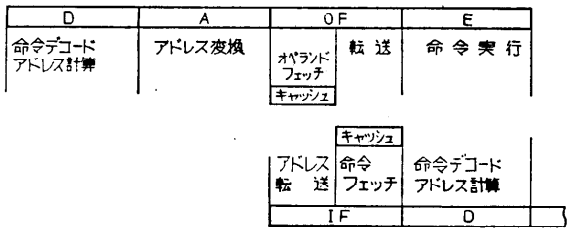

図-11　M-200H におけるキャッシュの半サイクル制御

は4Bの命令を8B単位で読出すため，2命令に1回出されている．図からわかるように，IF と OF（オペランド読出し）は2サイクルに1回重なっており，キャッシュが1マシンサイクルに1回しかアクセスできなければ図-10のような制御はできず，平均的には1.5マシンサイクル/命令となってしまう．このため，M-200H では図-11のように1サイクルに2回のキャッシュ・アクセスを可能とし，マシンサイクルの前半のアクセスはオペランド読出し，後半のアクセスは命令の読出しとストアに割当てている．

この制御を実現するためには高速アクセスのメモリが必要である．このため，アクセス時間が最大7nsの1K ビットのバイポーラ RAM を開発し，使用している[30]．

（2）　IA (Index Array) の採用

図-11からもわかるように，命令をデコードし，アドレスを計算してから，TLB とアドレスアレイを索引し，キャッシュからの読出しを開始するまでに1マシンサイクルしかない．このため，論理アドレスを TLB カラム・アドレスとキャッシュ・カラム・アドレスに分けて TLB，キャッシュのアドレスアレイの同時索引制御（パラレル——シリアル制御）を採用し，また，読出した登録アドレスと目的アドレスを高速に比較するために，メモリ・チップ上にアドレス比

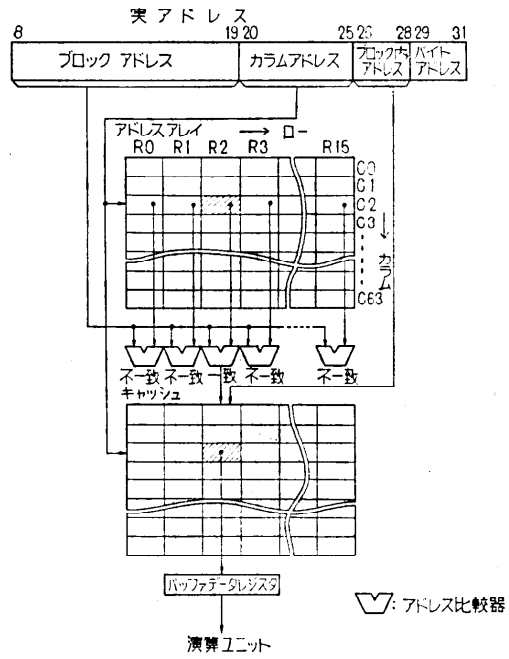

図-12　M-200H におけるキャッシュからの読出し制御

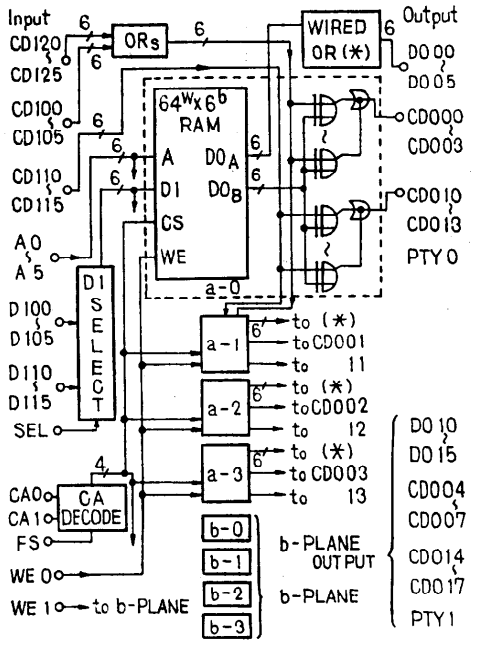

図-13　IA チップの構成図

較器を置く専用の LSI（これを Index Array——IA と呼んでいる）を開発した．IA は6ビット×64ワード×8組（3072ビット）のメモリと，それに対応する

比較器から構成され，TLB，アドレスアレイに共通に使用できるようになっている．キャッシュからのオペランド読出し制御を図-12に，またIAの構成を図-13に示す[45]．IAをアドレスアレイに使用する場合には，図-13のA0～5にカラム・アドレスを入力してデータ（登録されているブロック・アドレス）を読出し，CDI 00～05, 10～15にTLBから得られた実アドレス中のブロックアドレスを入力してこれと比較し，その結果をCDO 00～07, 10～17に得る．

以上に述べたように，実際にキャッシュを設計にするには，単に既存の各種方式の中から選択するだけでなく，個々の計算機に適した制御方式の改善，工夫が必要である．

5. 今後の展望

最後にキャッシュの今後の展望を考えてみたい．1章にも述べたように，キャッシュはMSのアクセス時間とCPUのマシンサイクル時間の差を埋める方式として考えられたものである．キャッシュを使用しなければ，かなり高度の先行制御を採用しても，MSのアクセス時間と同程度の平均命令実行時間しか実現できない．キャッシュの採用によって初めてMSのアクセス時間の数分の1の平均命令実行時間が得られたといってよい．

ところで，MS素子はコアから半導体への転換によってアクセス時間は格段に速くなり，また大容量化が可能となったが，今後は，現在のアクセス時間（素子レベルで速くても100 ns程度）のまま，大容量化の方向に進むと思われる．一方，CPUのマシンサイクルは高速化のためにさらに短縮されるから，今後は，MSのアクセス時間とCPUのマシンサイクル時間の差は大きく開き，キャッシュはますます重要な技術となろう．また，広がる一方の差を埋めるために，キャッシュとMSの間に，さらにもう1レベルの記憶階層を置くことも十分予想される（これをMSに対しWS——Work Storageと呼ぶことがある）．この場合，WSは各CPUが有するのではなく，MSのようにメモリシェアでアクセスされるであろう．

大型機でのキャッシュ容量は，現在64 KBが標準となっているが，より高密度のメモリ素子の開発[31]により，1 MB単位のキャッシュも近い将来に登場してこよう．

また，キャッシュ用メモリ素子については，より高速のアクセス時間が要求される一方，アドレスアレイ用の特殊高速素子もますます重要になろう．

6. おわりに

以上，キャッシュに関する構成，制御上の各種方式，M-200 Hにおける構成例，今後の展望などを述べた．このほか，キャッシュのバンク制御，キャッシュを採用した場合のコストの評価，キャッシュとプログラミングとの関係[38]なども重要な問題である．また，ここに述べたキャッシュの方式は，おもに大型機を対象としたが，最近はUNIVAC社の1100/60シリーズや，DEC社のVAX-11シリーズ[44]をはじめとするミニコンにまでキャッシュの技術が拡大している．

参考文献

1) 飯塚：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 および No. 8, pp. 540-547 (1972).
2) Gibson, D. H.: Consideration in Blockoriended System-design, SJCC, pp. 75-80 (1967).
3) Katzan, H. Jr.: Storage Hierarchy System, SJCC, pp. 325-336 (1971).
4) Meade, R. H.: On Memory System Design, FJCC, pp. 33-51 (1970).
5) Chow, C. K.: On Optimization of Storage Hierarchies, IBM J. Res. Develp., May, pp. 194-203 (1974).
6) Chow, C. K.: Determination of Cache's Capacity and its Matching Storage Hierarchy, IEEE, Vol. C-25, No. 2, pp. 157-164 (1976).
7) 元岡編：計算機システム技術，オーム社，pp. 24-40 (1973).
8) Censier, L. M.: A New Solution to Coherence Problems in Multicache System, IEEE, Vol. C-27, No. 12, pp. 1112-1118 (1978).
9) Mead, R. M.: How a Cache Memory Enhance a Computer Performance, Electronics, Vol. 45, No. 2, pp. 58-63 (1972).
和訳：日経エレクトロニクス，No. 32, pp. 100-109 (1972).
10) Tang, C. K.: Cache System Design in the Tightly Coupled Multiprocessor System, AFIPS, Vol. 45, pp. 749 (1976).
11) Liptay, R. M.: Structual Aspects of the System/360 Model 85; II The Cache, IBM Sys. J., Vol. 7, No. 1, pp. 15-21 (1968).
12) 最新超大型コンピュータの機能と技術，日経エレクトロニクス別冊「コンピュータ」，pp. 75-80 (1978).
13) 海野：IBM 3033 プロセサの内部設計とパフォーマンス，日経エレクトロニクス別冊「コンピュ

ータ」, pp. 85-96 (1978).
14) A Guide to the IBM System/370 Model 155, IBM Corp. GC 20-1729.
15) A Guide to the IBM System/370 Model 158, IBM Corp. GC 20-1754.
16) A Guide to the IBM System/370 Model 165, IBM Corp. GC 20-1730.
17) A Guide to the IBM System/370 Model 168, IBM Corp. GC 20-1755.
18) A Guide to the IBM 3033 Processor Complex of System/370, IBM Corp. GC 20-1859.
19) A Guide to the IBM 4341 Processor, IBM Corp. GC 20-1870.
20) HITAC M-200 H 機能説明書, 日立製作所, 8870-2-002.
21) HITAC M-180 機能説明書, 日立製作所, 8080-2-002.
22) 神田他: FACOM M シリーズのハードウェア, FUJITSU, Vol. 27, No. 4, pp. 19-45 (1976).
23) FACOM M-200 (カタログ), 富士通.
24) 国産新シリーズ・コンピュータの販売戦略と受注状況, 日経エレクトロニクス, 1976. 8. 23, pp. 150-157 (1976).
25) 西野: 超高性能電子計算機, 日立評論, Vol. 54, No. 3, pp. 49-51 (1972).
26) 関口: DIPS-1 システムの概要, 日立評論, Vol. 54, No. 3, pp. 52-57 (1972).
27) 高橋他: DIPS-1 L システム (ハードウェア), 日立評論, Vol. 54, No. 3, pp. 58-64 (1972).
28) 特許: メモリコントロールシステム, 特公昭 51-49535, IBM.
29) 特許: 多重データ処理システム, 特公昭 52-14064, IBM.
30) Hotta, A., et al.: Bipolar Memory LSI Chips for Computers, ISSCC DIGEST OF TECHNICAL PAPERS, pp. 98-99 (1979).
31) Inadachi, M., et al.: A 6 ns 4 Kb Bipolar RAM using Switched Load Register Memory Cell, ISSCC DIGEST OF TECHNICAL PAPERS, pp. 108-109 (1979).
32) 飯塚: スレーブ・メモリをもつ計算機システムの簡単な解析, 電子通信学会論文誌, Vol. 54, No. 8, pp. 698-705 (1971).
33) Mattson, R. L., et al.: Evaluation Techinique for Storage Hierarchies, IBM Sys. J., Vol. 9, No. 2, pp. 78-117 (1970).
34) Gecsei, J.: Determining Hit Ratio for Multilevel Hierarchies, IBM J. Res. Develop., pp. 316-327 (July 1974).
35) 中村他: バッファメモリ方式のシミュレーション, 情報処理, Vol. 15, No. 1, pp. 26-33 (1974).
36) 宮崎他: バッファメモリを持つマルチプロセッサの主メモリ競合の一解析法, 昭和 54 年電子通信学会情報システム部門全国大会講演論文集, pp. 340 (1979).
37) Spirn, J.: Distance String Models for Program Behavior, COMPUTER, Vol. 9, No. 11, pp. 14-19 (1976).
38) Denning, P. J.: The Working Set Model for Program Behavior, C. of ACM, Vol. 11, No. 5, pp. 323-333 (1968).
39) 飯塚他: 新しいアドレス・パタ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, 情報処理, Vol. 14, No. 9, pp. 669-676 (1973).
40) 飯塚他: 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, Vol. 37, No. 6, pp. 19-40 (1973).
41) 飯塚他: キャッシュ・メモリのシミュレーション, 電子総研彙報, Vol. 37, No. 8, pp. 19-37 (1973).
42) Ackland, B. D.: A Bit-slice Cache Controller, COMPCON '79, Spring, pp. 75-82 (1979).
43) Bell, J., et al.: An Investigation of Alternative Cache Organization, IEEE, Vol. C-23, No. 4, pp. 346-351 (1974).
44) Strecker, W. D.: Cache Memories for PDP-11 Family Computers, Proc. of the 3rd Annual Symposium on Computer Architecture, pp. 155-158 (1976).
45) 中澤他: LSI 技術の助けを借りてパイプライン方式を強化した最高速の商用汎用コンピュータ, 日経エレクトロニクス, No. 228, pp. 104-130 (1979).
46) 吉岡他: バイポーラ高速メモリのシステムへの応用, 電子科学, Vol. 30, No. 1, pp. 31-39 (1980).

(昭和 55 年 2 月 12 日受付)