解　説

メモリシステム技術†

高村守幸††　伊藤修二††

1. はじめに

メモリシステムには，いつの時代でも高速，大容量，低価格が要求されてきた．しかし高速メモリ素子ほどビット単価が高く小容量であるので，単一のメモリによりこれらの要求を満たすことはできない．異なった性能，特性を有する複数のメモリを組み合せて階層的にメモリシステムを構築する記憶の階層化は合理的なアプローチといえる．

またプログラマの立場からは，記憶容量の制約はプログラムを作成する上で厄介な問題をもたらす．すなわち，それはオーバレイなどの面倒な手法を用い，分割したプログラムを限られた記憶上で置換・ロードしなければならない．また，記憶容量の異なるシステム間でプログラムの互換性がとれないなどである．仮想記憶方式は以上のような記憶容量の制約から，プログラマを解放した．

このような，記憶の階層構成，仮想化の技術はメモリシステム技術の中で特に重要な技術である．ここでは，記憶階層及び仮想記憶方式の基本について述べる．

2. 記憶階層

2.1 概　説

記憶階層の技術とは，複数のメモリを組み合せて，メモリシステムの速度，容量，コストを最適化する技術である．現在のコンピュータシステムでは，CPUが直接アクセスする高速小容量のキャッシュメモリなどの内部記憶より，入出力チャネル装置を介してアクセスする低速大容量の磁気ディスクなどの外部記憶に至るまで複数レベルの記憶階層が形成されている．システムの処理性能を向上させるのに必要なことを，記憶階層の観点から考えると，

（1）記憶階層間での情報の移動回数を少なくする，

（2）記憶階層間での情報の移動速度を向上する，

の2点である．（1）については，CPUの参照頻度の高い情報，及び参照される可能性の高い情報を上位階層のメモリに置くように情報の移動を制御することにより情報の移動回数を少なくすることができる．また（2）については，より高速のメモリ素子を使用することにより各記憶階層での情報の移動速度を向上できるが，そのほかに，速度ギャップの大きい階層間には新たな階層を設けて情報の移動速度を向上させる手法も考えられる．

記憶階層間の速度ギャップについては主記憶と外部記憶，特に磁気ディスクとの間の速度ギャップが，大きい．磁気ディスクのアクセスタイムは主記憶のアクセスタイムの約10^4〜10^5倍であり，この間の速度ギャップを埋める新たな階層として現在では，半導体ディスク，ディスクキャッシュが実用化されている[1]．また半導体技術の向上によるCPUの高速化は，CPUと主記憶の間の速度ギャップの拡大をもたらし，この速度ギャップを埋めるために従来のキャッシュメモリと主記憶の間にさらに1段キャッシュメモリを追加した3階層メモリ方式も実用化されている[2)〜4)]．**図-1**に記憶階層の模式図を示す．

次に各記憶階層について個別に述べる．なお，本稿では外部記憶は半導体ディスク，ディスクキャッシュの説明までとし，磁気ディスク，磁気テープなどの磁気装置については述べない．

2.2 キャッシュメモリ

CPUのマシンサイクルと主記憶のアクセスタイムとの速度ギャップを埋めるために高速メモリをCPUと主記憶の間に置くキャッシュメモリ方式は，CPUの処理性能の向上を図るための常套手段となっている．キャッシュメモリが初めて実用化されたのは1968年にIBMが発表した360/85[5),6)]においてであり，それ以来各社の大型コンピュータに採用されてきたが，今やマイクロプロセッサにまで使用されるように

† Memory System Technologies by Moriyuki TAKAMURA and Syuji ITO (Kawasaki Works, Fujitsu Ltd.).

†† 富士通(株)川崎工場

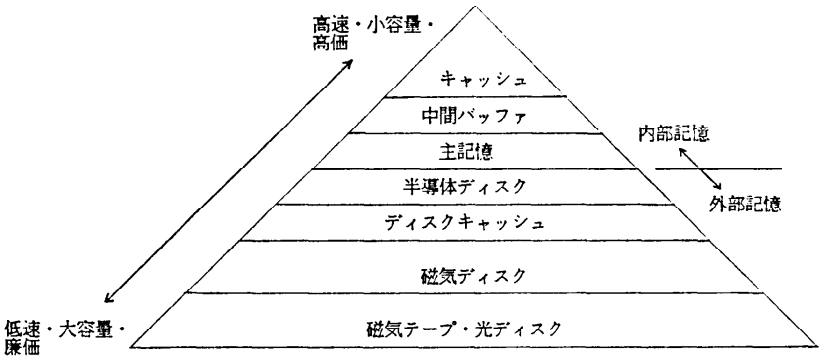

図-1 コンピュータシステムの記憶階層

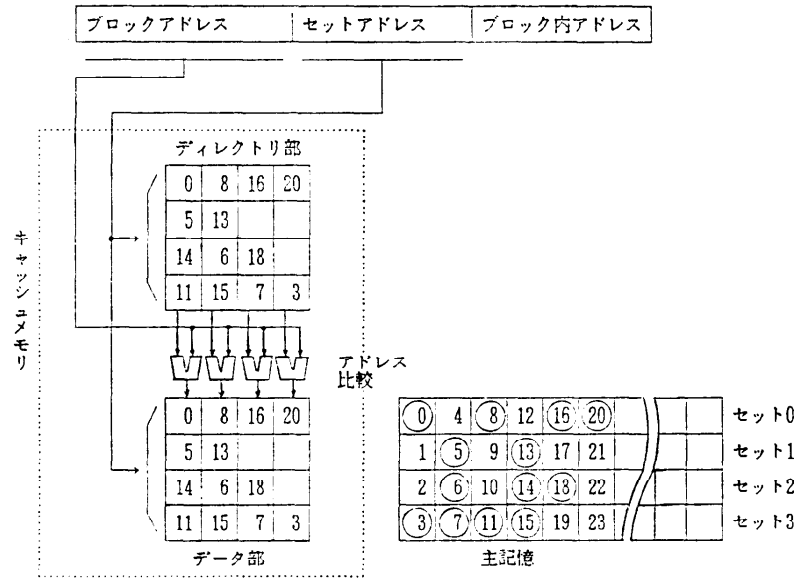

図-2 セットアソシアティブ方式

なった.

キャッシュメモリ方式は，よく知られているように「プログラムにおける参照の局所性」という現象を利用したものである．これは通常のプログラムの実行に際しては，短時間でみると，そのプログラムのメモリ参照はある一部のアドレス領域に集中しているという特性である．このことから主記憶の内容の一部をキャッシュメモリに複写しておき，CPU からのメモリアクセスの大部分をキャッシュメモリで処理することが可能となる．キャッシュメモリの容量は，当初は 8 kB 程度であったが，最近の大型コンピュータでは 128〜256 kB と大容量化し，システムの高性能化を図っている．次にキャッシュメモリ方式の制御技術について述べる．

（1） 主記憶とのマッピング方式

主記憶及びキャッシュメモリを同一サイズのブロック（通常 64〜128 B）に分割し，ブロック単位でマッピングを行う．マッピングの方式にはいくつかの方式があるが，キャッシュメモリのヒット率の良さ，制御回路のハードウェア量が少ないことから，現在ではセットアソシアティブ方式が大部分のコンピュータに採用されている．セットアソシアティブ方式の概要を図-2 に示す．この方式では主記憶及びキャッシュメモリを，複数のブロックで構成されたセットにグループ化し，マッピングは同一セット間で行う．ディレクトリ部には対応するデータ部のブロックに主記憶のどの

ブロックの情報が入っているかを示すアドレス情報を入れておき，メモリがアクセスされたとき，ブロックアドレスと，セットアドレスで指定されたディレクトリ部のアドレス情報を比較し，一致したブロックのデータを読み出す．主記憶のあるセットに属するブロックは，キャッシュメモリの対応するセットの任意のブロックにマッピングされるが，このときにキャッシュメモリにブロックの空きがない場合は，そのセットの中よりあるブロックを選択し，キャッシュメモリより主記憶に追い出して，空きをつくる必要がある．追い出すブロックを選択するアルゴリズムとしては，LRU 方式 (Least Recently Used) が一般的である．LRU 方式は，そのセットの中のブロックで最も以前に参照されたブロックを追い出す方式である．

（2） ストア方式

主記憶への書込み方式には，ストアスルー方式及びストアイン方式の二方式がある．

ストアスルー方式は主記憶への書込み要求に対して，キャッシュメモリとともに主記憶も同時に書換える方式である．この方式では，書込み時には必ず主記憶アクセスが生じるが，キャッシュメモリと主記憶の内容は常に一致している．

これに対して，ストアイン方式は主記憶への書込み要求に対して，キャッシュメモリのみ書換えを行い，主記憶の書換えは行わない．このとき，主記憶のデータは古くなっているが，主記憶の書換えはその後ブロックの置換が起こったときに行われる．

主記憶を共有するマルチプロセッサを構成するときには，システムに存在する複数のキャッシュメモリの内容を一致させる制御を行う必要がある[7),8)]．ストアスルー方式では更新されたデータが常に主記憶に存在するので，制御は比較的容易である．すなわち，あるCPU のキャッシュメモリに書込みが生じた場合，その CPU は主記憶に対し，書込み動作を行う．主記憶はほかの CPU のキャッシュメモリにそのブロックが存在していた場合，そのブロックの無効化をそのCPUに指示する．

一方，ストアイン方式では，ある CPU のキャッシュメモリに書込みが生じた場合，ほかの CPU の該当するブロックは無効化されなければならない．この時点で，更新されたデータはその1台の CPU のキャッシュメモリ上だけに存在するため，ほかの CPU がそのデータをアクセスする場合には，更新されたデータを主記憶経由で持ってこなければならない．このように，ストアイン方式はストアスルー方式に比べ制御が複雑である．

2.3 3階層メモリ

半導体技術及び実装技術の向上により CPU のマシンサイクルは 5〜10 ns 前後にまで高速化された．一方，主記憶のメモリ素子として一般的に使用される DRAM は，集積度の向上に主眼を置き，アクセスタイムの改善は小さいため，CPU と主記憶との間の速度ギャップは拡大している．一般にキャッシュメモリを有する CPU の性能は次式で近似できる[2)]．

$$T_{avg}=T_c+\alpha\cdot T_{acc} \qquad (1)$$

ここで T_{avg} は CPU の平均命令実行時間，T_c はキャッシュメモリ上に命令，オペランドが存在するときの CPU の平均命令実行時間，T_{acc} は CPU から見た主記憶のアクセスタイムである．またαは求める命令，オペランドがキャッシュメモリ上に存在しない確率（キャッシュミス率）である．(1)式から，T_{avg} を小さくして CPU の処理性能を向上するためには，

① T_c を小さくする，

② α を小さくする，

③ T_{acc} を小さくする，

が必要である．①のためには，CPU のマシンサイクルタイムを短縮すること，及び命令実行に要するサイクル数を減少することが必要である．

②を実現するためには，キャッシュメモリの容量を大きくすることが一番効果的である．しかし，キャッシュメモリの容量を大きくすると物理的サイズも大きくなり，信号の遅延時間が増大し，マシンサイクルに影響するため，最適点が存在する．③は CPU と主記憶の間の速度ギャップを埋めようとするものであり，その手段としては，

（i） DRAM に換えて，主記憶に高速メモリを使用する2階層方式，

（ii） キャッシュメモリと主記憶の間に中間バッファメモリを置く3階層メモリ方式，

などが考えられる．

汎用コンピュータにおいて，3階層メモリ方式は，1981 年に発表された富士通の M-380 で初めて実用化された．その後，1985 年に発表された，日電，日立の最上位コンピュータに採用されている．**図-3** に3階層メモリ方式の構成図を，**表-1** に各社の3階層メモリの記憶容量を示す[2)-4)]．

次に3階層メモリ方式による CPU 処理性能向上について述べる．3階層メモリ方式をとる CPU の性能

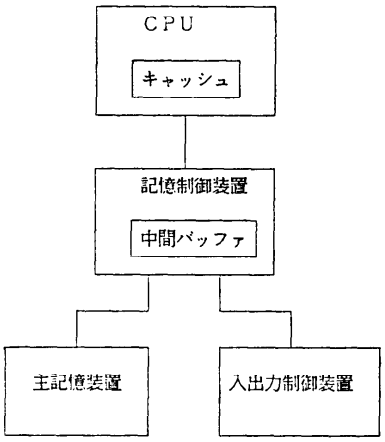

図-3 3階層メモリの構成

表-1 大型コンピュータにおける3階層メモリ

	富士通 M-380	日電 ACOS 1510	日立 M-680H
キャッシュ容量	64 kB	128 kB	256 kB
中間バッファ容量	256 kB	512 kB	1 MB
主記憶容量	128 MB	128 MB	256 MB

は次式で近似することができる.

$$T_{avg}=T_c+\alpha\cdot T_{acc'}+\alpha'\cdot(T_{acc}-T_{acc'}) \quad (2)$$

$T_{acc'}$ は中間バッファメモリのアクセスタイム, α' は求める情報がキャッシュメモリにも中間バッファメモリにもない確率である. 中間バッファメモリを主記憶の一部と見た場合, 見かけ上の主記憶のアクセスタイムを $T_{acc''}$ とすると

$$T_{avg}=T_c+\alpha\cdot T_{acc''} \quad (3)$$

(2), (3)より

$$T_{acc''}=T_{acc'}+\frac{\alpha'}{\alpha}\cdot(T_{acc}-T_{acc'}) \quad (4)$$

となる. キャッシュメモリの容量を 64 kB, 中間バッファメモリの容量を 256 kB とするとプログラムの種類にもよるが, α'/α は通常 1/4〜1/8 となる. 中間バッファメモリのアクセスタイム $T_{acc'}$ を主記憶のアクセスタイム T_{acc} の 1/5 とし, α'/α を 1/4 とすれば,

$$T_{acc''}=\frac{2}{5}\cdot T_{acc} \quad (5)$$

となり, 主記憶のアクセスタイムは実効的に 2/5 に短縮されたことになる.

2.4 主記憶

大規模オンライン・システムなどの巨大システムでは, 高処理能力を実現するために大容量の主記憶が必要である. このため主記憶は, 高速化よりむしろ大容量化を指向してきた. 主記憶には通常 DRAM が使用されているが, DRAM の集積度は3年に4倍のペースで向上しており, それに応じて主記憶も拡張してきた. 現在では 256 Kbit の DRAM が広く実用されており, さらに 1 Mbit の DRAM の実用も開始された. 図-4 に大型コンピュータの最大主記憶容量の変遷を示す.

主記憶のスループットを向上させる手法としてインタリーブ方式が効果的である. インタリーブ方式は, 主記憶を独立に動作可能な複数のバンクに分けて構成し, マシンサイクルごとにバンクをリード/ライトして主記憶全体のスループットを向上させるものである. この方式は制御回路の物量が大きくなるため, 従来は高性能を追求する大型コンピュータに使用されていたが半導体技術の向上により制御回路の LSI 化が容易となり, 小型コンピュータにも適用されるようになった.

3階層メモリ方式は新たな階層の追加により主記憶のアクセスタイムを実効的に短縮するものであった. 一方, 2.3 節の項(i)に示した, 高速メモリ素子を用いた主記憶も実用化されている. アクセスタイムが DRAM に比べ2〜3倍も高速な SRAM を主記憶に使用し, アクセスタイムを短縮して処理性能の向上を図った汎用コンピュータとして, 富士通が 1985 年に発表した, 大型コンピュータ M-780 がある[9]. SRAM

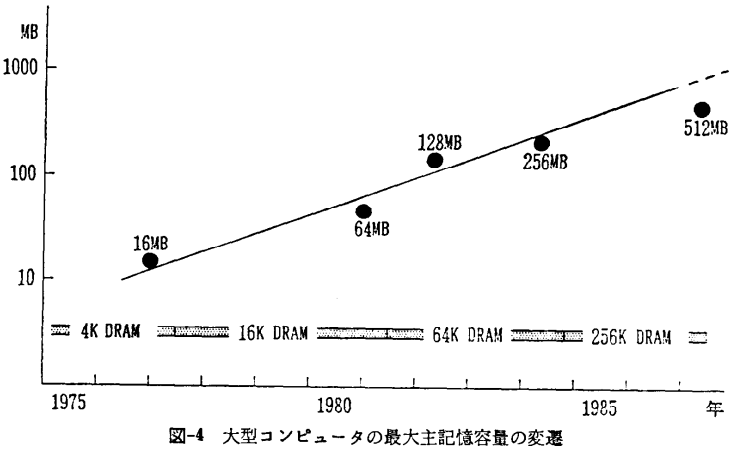

図-4 大型コンピュータの最大主記憶容量の変遷

を用いた主記憶による2階層メモリ方式と，中間バッファを挿入した3階層メモリ方式との性能の優劣は一概には言えないが，どちらの方式を採用するかは性能，コスト，ハードウェア量などを勘案して決定する必要がある．どちらも CPU，主記憶間の速度ギャップを埋めて高処理能力を実現する手段である．

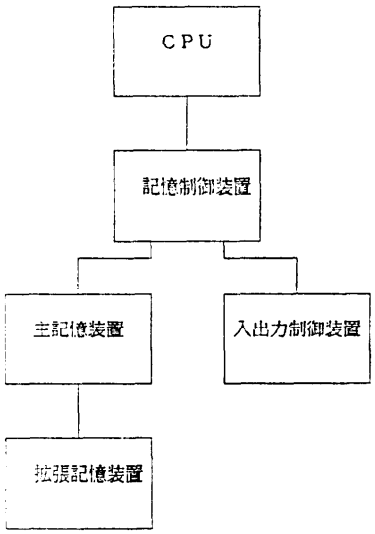

図-5 拡張記憶を備えたシステム

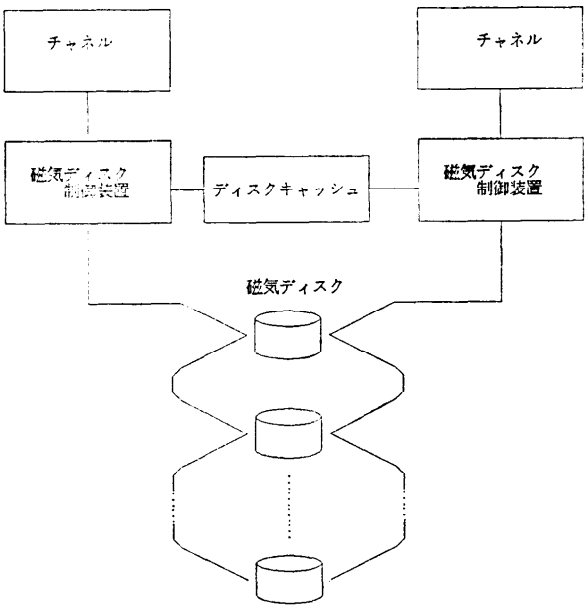

図-6 ディスクキャッシュ・システムの構成

2.5 拡張記憶

主記憶空間を越えるデータは通常磁気ディスクなどの外部記憶に格納される．外部記憶のアクセスタイムは CPU の処理時間に比べ極端に遅いのが普通であるので，外部記憶のデータの入出力命令のオーバヘッドはシステムの処理性能を低下させる．拡張記憶は主記憶との間で直接データ転送を高速に行う大容量記憶である．図-5 に拡張記憶を装備したシステムの模式図を示す．取り扱うデータ量が多量で，入出力命令を用いてデータへのアクセスが頻繁に行われる場合に有効であり，拡張記憶はスーパ・コンピュータや汎用の大型コンピュータにおいて実用化されている[10),11)]．

2.6 半導体ディスク，ディスクキャッシュ

磁気ディスクと主記憶の間には前述したように大きな速度ギャップがある．主記憶のアクセスタイムがせいぜい数 100 ns に対して磁気ディスクのアクセスタイムは数 10 ms であり，この速度ギャップがシステムの処理性能向上の一つのネックとなっている．半導体ディスク，ディスクキャッシュは，この速度ギャップを埋めるものである．

半導体ディスクは記憶素子に DRAM を使用した外部記憶であり，磁気ディスクと同様に使用することができる．磁気ディスクに比べヘッドの移動，回転などの動作が不要なため，磁気ディスクよりはるかに高速のアクセスが可能であり，またアクセスタイムは全領域均一である．アクセスタイム 0.3 ms，データ転送速度 3 MB/sec の装置が実用化されている．半導体ディスクには参照頻度の高いデータセットを格納しておくことにより，その効果を高めることができる．

ディスクキャッシュは CPU，主記憶間のキャッシュメモリ方式を主記憶と磁気ディスク間に適用して磁気ディスクの実効的なアクセスタイムを短縮し，システムの処理性能の向上を図るものである．本体側に組み込む形態のものと，磁気ディスク制御装置側に付加する形態のものがあるが，現在では，磁気ディスク制御装置側に付加する形態のものが一般的である．ディスクキャッシュはメモリ素子として DRAM を使用しており，記憶容量は 4〜64 MB のものが実用化されている．図-6 にディスクキャッシュ・システムの構成図を示す．

3. 仮想記憶方式

3.1 概　説

現在のコンピュータシステムではシステム資源の有効利用を図るため，多重プログラミングがサポートされているのが普通である．主記憶装置を仮想化しないで実記憶で多重プログラミングをサポートしようとすると，以下の問題点がある．

（1） 主記憶の断片化の問題

主記憶上で複数のプログラムを実行する際，各プログラムの容量，終了のタイミングはまちまちであり，また，割り付けられた記憶域はそのプログラムが終了するまで変らないので，プログラムの入替えに伴い主記憶内に未使用の領域が散在するようになる．このとき新たにプログラムをロードしようとしてもプログラムの容量によってはどの未使用領域もプログラムの容量より小さいためロードできない事態が生ずることがある．このように，主記憶の未使用領域が断片化して遊休化する問題がある．

（2） 主記憶の利用効率の低下

プログラムが実行されるとき，参照される部分はプログラムの一部分であり，常にプログラムの全体が同時に必要なわけではない．不要な部分までを主記憶にロードすることは主記憶の利用効率を下げることになる．

（3） プログラムが使用できる主記憶領域の容量制限

プログラムの容量が大きくなるにつれて，プログラムが必要とする領域の容量を確保することが困難になるため，プログラマはオーバレイによるプログラミングをしなければならず，プログラマの負担が大きい．

以上のような問題点は仮想記憶方式を導入することにより改善することができる[12)~15)]．

仮想記憶は磁気ディスクなどの大容量外部記憶を主記憶のバックアップとして持ち，論理アドレス空間を外部記憶上に展開しておき，プログラム実行時に必要とする部分を，順次外部記憶より主記憶に移動し，プログラムがほとんど主記憶上で実行されているようにするものである．主記憶と外部記憶との間の情報の移動は主にオペレーティングシステムが制御している．こうして，プログラマから見えるメモリを，物理的な主記憶の持つ物理容量（番地）の制限から分離独立させ，プログラマが実際に利用できる容量（番地）を論理的限界一杯まで拡張するものである．

仮想記憶方式を実現するのに，ページング方式及びセグメンテーション方式の二つの方式がある．ページング方式は論理アドレス空間を固定長のページと呼ばれる単位に分割する方式であり，主記憶と外部記憶の間での情報の移動はページ単位で行われる．論理アドレスより物理アドレスへの変換はページテーブルと呼ばれる変換テーブルを参照して行われる．

セグメンテーション方式ではセグメントと呼ばれる論理的に意味を持つひとかたまりの可変長情報の単位を設定し，セグメント単位で情報の移動を行う．どちらの方式も主記憶と外部記憶の間での情報の移動についてはオペレーティングシステムが自動的に管理していて，プログラマは関知しなくて済むため，プログラマには仮想的な大容量の主記憶が提供されたように見える．このため，この方式を仮想記憶方式と呼んでいる．ページング方式及びセグメンテーション方式については後に詳述する．次に仮想記憶を実現する上で必要となる制御技術について述べる．

3.2 制 御 技 術

（1） 動的アドレス変換機構 (DAT, Dynamic Address Translation)

論理アドレスを，命令実行時にアドレス変換テーブルを参照して物理アドレスへ動的に変換する機構である．アドレス変換速度の観点からハードウェアで用意されているのが普通である．アドレス変換テーブルは主記憶内に存在しており，アドレス変換ごとに主記憶アクセスが生じることになると，システムの処理性能の低下を招く．このため，通常は CPU 内に高速メモリにより構成されたアドレス変換用高速バッファを持ち，そこに最近参照された論理ページと物理ページの変換対を保持し，変換の必要が生じたとき，この対応表を参照し，求める変換対が存在した場合にはこの表より高速に物理ページを求める．このようにして主記憶へのアクセスを減らし，アドレス変換によるオーバヘッドを小さくしている．

（2） アドレス変換割り込み制御

求めるページ，あるいはセグメントが主記憶内に存在しない場合には DAT によるアドレス変換の途中でページフォールトまたはセグメントフォールトと呼ばれる割り込みが発生する．この場合には制御はオペレーティングシステムの制御プログラムに移り，制御プログラムでは，求めるページ，あるいはセグメントを外部記憶より主記憶内の空領域へ転送する．この動作をページインあるいはセグメントインと呼ぶ．主記

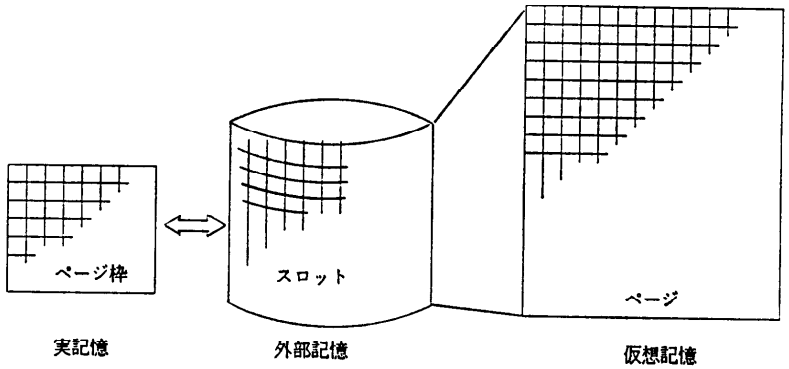

図-7 ページングにおける外部記憶と仮想記憶

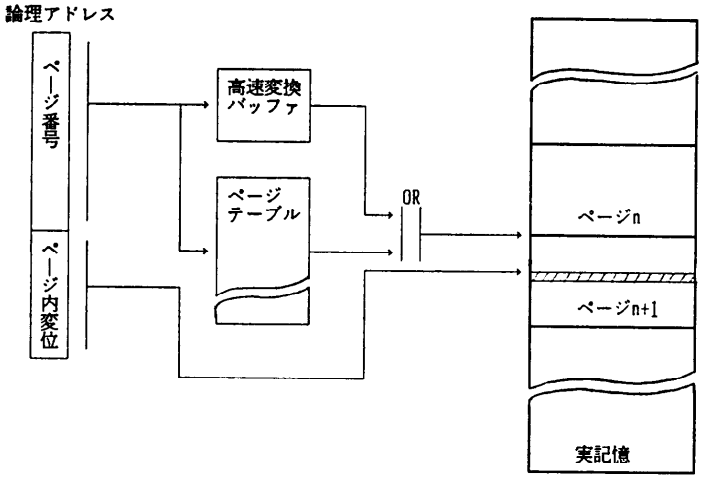

図-8 ページング方式におけるアドレス変換

憶にあるアドレス変換テーブルは同時に更新される.

(3) 主記憶管理

ページインあるいはセグメントインのとき主記憶内に必要とする大きさの空領域が存在しない場合には制御プログラムは主記憶内のいずれかのページあるいはセグメントを外部記憶へ追い出す. この動作をページアウトあるいはセグメントアウトと呼ぶ. 空領域を確保して, 要求されているページあるいはセグメントを主記憶内に取り込む (ページイン, セグメントイン). この置換対象とするページあるいはセグメントの選択アルゴリズムはシステムの処理性能に大きな影響を与える. 基本的には近い将来最も参照されないページあるいはセグメントを予測して置換対象を選択することになるが, この選択についていくつかのアルゴリズムが提案されている. 良く知られているアルゴリズムにLRU, FIFO (First In First Out), RANDOM アルゴリズムがあるが, LRU アルゴリズムがすぐれている. LRU アルゴリズムは主記憶内のページあるいはセグメントのうち最も遠い過去に参照されたものを置換対象とする. これは最近参照されていないページあるいはセグメントは近い将来においても参照される確率が小さいという特性に基づいている.

3.3 ページング方式

ページング方式とは図-7 に示すように仮想記憶が有する論理アドレス空間を固定長のページとよばれる区画に分割する方式である. 外部記憶, 主記憶も同様に分割する. 外部記憶の区画をスロット, 主記憶の区画をページ枠と呼ぶ. ページ枠のサイズは普通 2kB または 4kB 程度の大きさがとられる. アドレス変換テーブルであるページテーブルには, 各ページの主記憶ページ枠への割り当て状況を示す, ページフォールトビット及びページ枠番号が記されている. 論理アドレス空間へのアクセスはプログラムが発生した論理アドレスにより行われる. 論理アドレスはページ番号及びページ内変位により構成されている. ページ番号はDAT によりそのページが主記憶に割り付けられているかを, ページフォールトビットにより調べられ, 割り付けが行われていた場合には対応するページ枠を読み出し, ページ内変位と合成されて実アドレスへの変換が行われる. 割り付けが行われていなかった場合にはページフォールト割り込みが発生し, オペレーティングシステムの制御のもとで必要なページのページイン, ページアウトが行われる. 図-8 にページング方式によるアドレス変換の概要を示す. ページング方式ではプログラムの実行に際しては, プログラムは

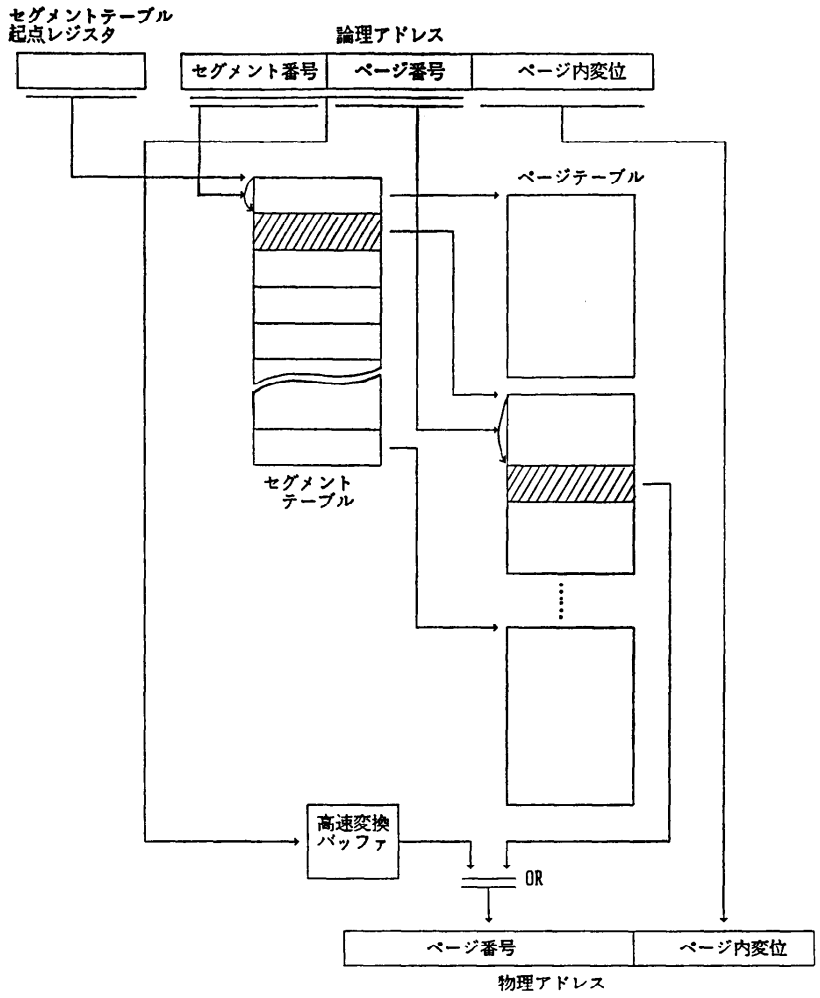

図-9 2レベル・ページング方式におけるアドレス変換

ページ単位で主記憶にロードされるので，実記憶方式での記憶の断片化，主記憶の利用効率の低下，の問題は改善される．

現在実用化されているページング方式では，領域管理を容易にし，かつページテーブルが必要とする主記憶の連続領域を少なくするなどの理由から，一定の複数ページにより構成されたセグメントと呼ぶ単位を導入している場合が多い[16]．論理アドレスはセグメント番号，ページ番号，ページ内変位に区分され，アドレス変換はセグメントテーブル及びページテーブルの二つの変換テーブルを使用して2段階に行われる．セグメントサイズは64kBないしは1MBがある．この方式は2レベル・ページング方式と呼ばれている．2レベル・ページング方式ではアドレス変換が2段階になるためアドレス変換のオーバヘッドが大きくなるが，この欠点は，前述したアドレス変換用高速バッファの使用により改善される．図-9に2レベル・ページング方式におけるアドレス変換の概要を示す．

3.4 セグメンテーション方式

セグメンテーション方式における論理アドレス空間は複数個のセグメントにより構成されている．セグメントは，たとえば配列，サブルーチンなど論理的な意味を持ったひとかたまりの情報に対し割り当てられる．このためセグメントサイズは一定ではない．論理アドレスはセグメント番号及びセグメント内変位に区分される．アドレス変換のためのセグメントテーブル

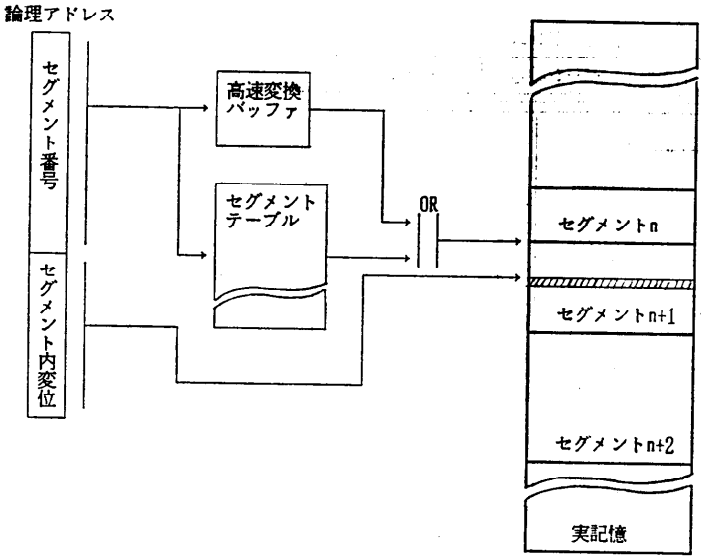

図-10 セグメンテーション方式におけるアドレス変換

はセグメントの個数だけのセグメント記述子により構成される．セグメント記述子が持つ情報としては，主記憶内のセグメントの存否を示すセグメントフォールトビット及びセグメントの主記憶上の先頭アドレス，セグメントの容量，セグメントに対するアクセス権などの情報である．これらの情報によりセグメント単位での情報の保護，共用などの制御が可能となる利点がある．図-10にセグメンテーション方式によるアドレス変換の概要を示す．セグメンテーション方式は情報の管理手段としては有用であるが，セグメントサイズが一定でないことに起因する主記憶の断片化の問題が発生する．これは前述した，実記憶方式下での多重プログラミングで問題となった主記憶の断片化と同様の事情により発生する．

セグメンテーション・ページング方式はセグメンテーション方式における記憶管理の手法にページング方式を導入したものである．個々のセグメントをページ化し記憶管理の単位としてのページを導入する．この場合アドレスはセグメント内変位がページ番号とページ内変位に分離される．各セグメントはページテーブルを持つことになり，アドレス変換はセグメントテーブル及びページテーブルの2段階で行われる．この方式により主記憶の断片化は改善される．

3.5 多重仮想記憶方式

以上述べた仮想記憶方式はシステム全体で1個の論理アドレス空間を保有している，単一仮想記憶方式である．この方式では1個の論理アドレス空間を複数プログラムが分割使用するため，個々のプログラムに割り当てられる記憶領域や多重度は制約を受ける．また各プログラム間相互での影響を完全には排除できないため，記憶保護が完璧でないという問題がある．

多重仮想記憶方式はプログラムごとに独立した1個の論理アドレス空間を与えるものである．このためにアドレス変換テーブルを複数個設け，それらを切り換えて使用することにより複数の論理アドレス空間を作り出している．論理アドレス空間の多重度はアーキテクチャによる制限がないため，理論上は多重度をいくらでも上げることができる．しかし実際には主記憶及び外部記憶の容量，主記憶と外部記憶との間の情報転送速度などの制約から最適の多重度が決定される．また1個のプログラムは1個の論理アドレス空間を占有使用できるため，単一仮想記憶方式での記憶領域の有限性によってもたらされた各プログラムの割り当て領域の制約はほとんどなくなる．さらに各プログラムがそれぞれ異なる論理アドレス空間で処理され，また各論理アドレス空間相互の独立性が保証されていることから，プログラム間相互の影響がなくなり，完全な記憶保護が実現される．このため，現在のシステムでは，多重仮想記憶方式が普通である．

4. 将 来 展 望

3階層メモリ，拡張記憶，半導体ディスク，ディスクキャッシュなど多様なメモリ階層を導入することにより，メモリシステムとしての最適化・効率化を一層進めようとしていることが，いままでの記述から理解されよう．このような新しい階層構造は，目覚しい進歩を遂げている半導体メモリ素子の大容量化，低価格化に支えられて初めて可能となったことに留意する必要がある．新素材素子など無限の可能性を持つ半導体素子の新しい発展をタイムリに取り入れたメモリシステムの構築が望まれる．そして，高速メモリ素子ほど，高価で小容量であることは今後も変らないであろうから，記憶階層はますます多レベル化，多様化すると考えられる．

仮想記憶により，実記憶の限界は拡張されたが，コ

ンピュータのメモリシステム全体としては，アドレスの方法，アロケート方法，オペレーション機能，データの保持性に不連続と断層が厳として存在している．

たとえば，主記憶にあるデータは，指定されたレジスタの内容が示すアドレスに変位を加えて生成されるアドレスによりアクセスされるが，磁気ディスクにデータが存在する場合は，デバイス番号，シリンダ番号，トラック番号，レコード番号，レコード内変位などでアクセスし，両者のアクセス方法はまったく異なっている．

さらに，加算，比較，移動などのオペレーションが，外部記憶を含めたメモリ・トータル・システムで共通には定義されていない．したがって，加算を実行しようとすると，データが記憶されている記憶媒体がなにかということを意識しなければならない．つまり，データが主記憶にあれば話は簡単である．データが，磁気ディスクにあると，上述のようにアドレス方法が異なるのみならず，プログラムは一連のチャネル・コマンド語（これもプログラム）を生成してデータを主記憶に移動しなければ加算ができない．これは，基本オペレーションがメモリ階層のある部分階層にしか定義されていないことに起因している．

さらにまた主記憶にあるデータは処理が終了すると通常は消滅するが，外部記憶のデータは陽に壊さない限りは保持されるという違いをプログラマは意識しなければならない．

要するに，記憶媒体のテクノロジとインプリメンテーションに依存した不連続性を意識したプログラミングをプログラマは余儀なくされているのが現状である．プログラム開発コストのうちの，かなりの割合がプログラマが記憶階層でのデータの管理を行うことと繁雑な入出力処理を行うことに費やされていると言われている．今後の巨大データベースシステムの開発を効率化・合理化するためには，記憶階層とそのデータ管理は最も重要な課題である．

この一つの解がワンレベル記憶である[17)~21)]．ワンレベル記憶はすべての階層の物理的な記憶装置は同一のアドレシング方法，基本オペレーション，データ保持によりアクセスされるようにメモリシステムを統合するものである．

前述のとおり，物理的記憶階層はますます多様化・多レベル化されようが，記憶階層トータルの中でのデータの移動・管理はプログラマの仕事からオペレーティングシステムとハードウェアの責務へ移行するものであるとも言える．

ワンレベル記憶により得られるメリットはきわめて大きい．その主要なものを挙げると，

① プログラムの生産性向上

入出力動作の概念がプログラムから全廃されるわけではないが，複雑な入出力動作がプログラムからなくなることにより開発効率化が図れる．

② テクノロジからの独立

記憶階層を構成する記憶媒体がなんであるかが見えなくなる（階層自体がない）ので，システム間でのプログラムのポータビリティが増す．磁気バブル，不揮発性半導体素子，光ディスクなどの記憶装置のテクノロジの将来は予測ができない程急激に変化している．記憶階層の数や記憶媒体の種類を変える新しい記憶装置を導入しても，既製のプログラムに変更を与えない．

ワンレベル記憶は，主記憶レベルに統合された，とてつもない大仮想空間（通常は48または64ビットアドレス）をサポートするので，実記憶へのアドレス変換は，従来以上に効率化する必要がある．しかし，ソフトウェアの生産性・ポータビリティの利点を考えると，ワンレベル記憶は将来ますます実用されるものと考えられる．

5. おわりに

CPUが，CPUアドレスにより直接アクセスする内部記憶においても，また，入出力命令によりチャネルを介してアクセスする外部記憶においても，各種のメモリを組み合せて階層構造を形成していること，それにより，より高速で，より大容量で，より低価格のメモリシステムを構築しようとしていることを述べた．

さらに，磁気ディスクなどの外部記憶でバックアップすることにより，仮想的に論理アドレス一杯まで主記憶容量を拡張して，実記憶容量限界を突破しようとしていることを述べた．

さらに，厳として存在する主記憶と外部記憶という区別を撤廃して，主記憶と同レベルの大アドレス空間として統合するワンレベル記憶の将来性にも言及した．

参考文献

1) 金子：汎用大型機のディスク・アクセスを高速化するディスク・キャッシュ方式と性能評価，日経エレクトロニクス，No. 364, pp. 205-232 (1985).
2) 平栗他：3階層メモリー方式や高密度化技術に

より性能向上を図った大型コンピュータ M-380/382, 日経エレクトロニクス, No. 276, pp. 176-199 (1983).
3) 馬場他：2レベルのキャッシュやパイプライン処理の工夫で速度を上げた大型コンピュータ ACOS 1500, 日経エレクトロニクス, No. 373, pp. 233-279 (1985).
4) 小高他：演算パイプラインや3階層記憶により高速化を図った M-680/682H の処理方式, 日経エレクトロニクス, No. 382, pp. 228-267 (1985).
5) Conti, C. J., Gibson, D. H. and Pitkowsky, S. H.: Structural Aspects of the System/360 Model 85; I General organization, IBM System Journal, Vol. 7, No. 1, pp. 2-14 (1968).
6) Liptay, J. S.: Structural Aspects of the System/360 Model 85; II The cache, IBM System Journal, Vol. 7, No. 1, pp. 15-21 (1968).
7) Tang, C. K.: Cache System Design in the Tightly Coupled Multiprocessor System, AFIPS Proc., Natl. Comput. Conf., 45, pp. 749-753 (1976).
8) Censier, L. M. and Feautrier, P.: A New Solution to Cache Coherence Problems in Multicache System, IEEE Trans. Comput., C-27, 12, pp. 1112-1118 (Dec. 1978).
9) 処理速度が M-380 の約2.2倍の M シリーズ, 最上位機種 M-780, 日経エレクトロニクス, No. 383, pp. 92-94 (1985).
10) 小高他：最大性能が 630 MFLOPS で1G バイトの半導体拡張記憶が付くスーパーコンピュータ HITAC S-810, 日経エレクトロニクス, No. 314, pp. 159-184 (1983).
11) 古勝他：最大性能 1.3 GFLOPS, マシン・サイクル 6 ns のスーパコンピュータ SX システム, 日経エレクトロニクス, No. 356, pp. 237-272 (1984).
12) 元岡編：計算機システム技術, オーム社, pp. 41-66 (1973).
13) 高橋, 土居, 益田：オペレーティング・システムの機能と構成, 岩波講座情報科学 16, 岩波書店, pp. 197-244 (1983).
14) 武井：仮想記憶方式, 情報処理, Vol. 21, No. 4, pp. 358-368 (Apr. 1980).
15) Langdon, G. G. Jr.: Computer Design, Computeach Press Inc., pp. 255-267 (1982).
16) 江村：オペレーティング・システムへの構造的アプローチ(下), 日本コンピュータ協会, pp. 931-1082 (1985).
17) 宇都宮：システム/38 の新しい思想とアーキテクチャ, bit, Vol. 15, No. 11, 共立出版, pp. 45-50 (1983).
18) Glenford, J. M.: Advances in Computer Architecture, A Wiley-Interscience Publication pp. 91-98 (1982).
19) NE レポート：単一レベル記憶を実現し, データベース機能を組み込んだ IBM システム/38, 日経エレクトロニクス, No. 201, pp. 56-63 (1978).
20) Houdek, M. E. and Mitchell, G. R.: Hash Index Helps Manage Large Virtual Memory, Electronics, Vol. 52, No. 6, pp. 111-113 (1979).
21) Durniak, A.: System/38 Shows How IBM Aims to Keep Up with the Times, Electronics, Vol. 52, No. 6, pp. 102-105 (1979).

(昭和61年4月3日受付)