計算機アーキテクチャ講義
ノート４
(3.3　メモリ構成　P.108～P.128)

平成19年12月３日配布

今瀬　真

メール：imase@ist.osaka-u.ac.jp
http://www.ispl.jp/~imase/lecture/comp-arch/2006/

HTテクノロジPentium 4の説明から抜粋

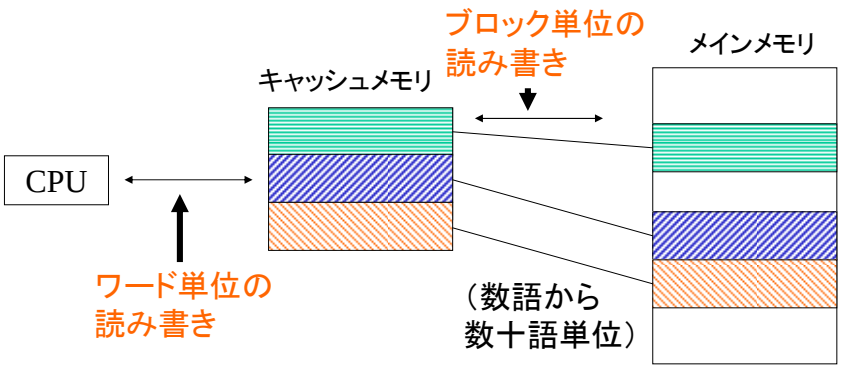
- 現代のマイクロプロセッサの多くは、1クロック・サイクルに複数の命令（インストラクション）を処理可能なスーパースカラー型のアーキテクチャを採用している。逆にいえば、スーパースカラー型マイクロプロセッサの実行ユニットは、同時に複数の命令処理が可能なように複数の演算器を内蔵しているのだ。通常は、1つのスレッドの中で、命令同士の依存関係の有無や分岐予測に従って、複数の命令を並列に処理する。つまり、インストラクション・レベルでの並列処理が行われる。
- しかし実際には、依存関係により並列処理ができなかったり、メモリからのデータの読み出し／書き込みを行う時間待ちがあったりするため、実行ユニットに内蔵される演算ユニットがフル稼働することは珍しい。Intelのホワイトペーパー（Introduction to Hyper-Threading Technology:Intel文書番号250008-002）によると、Pentium 4プロセッサの実行ユニットの使用率は35%に過ぎないという。

3.3.1　キャッシュメモリ

- 一般的に、メインメモリに使われているDRAMのスピードはCPUに比べてかなり遅く、CPUの命令実行速度を下げる原因となっている。
- この問題を解決するために、CPUとメインメモリの間にキャッシュメモリと呼ばれる高速／小容量のメモリを配置する。
- メインメモリからキャッシュメモリへはブロック（数語から数十語単位）にまとめて転送する手法がとられる。
- メモリシステムの高速化のために、キャッシュメモリが2段、3段と重ねて実装されることがある。この場合、CPUに近い位置にあるほうから1次キャッシュ、2次キャッシュ…と呼ばれる。
- 現状ではCPUチップ内に、1K～16Kbytes程度の1次キャッシュを内蔵していることが多い。
- また現在のPC互換機では、高速SRAMを用いて64K～1Mbytes程度の2次キャッシュを実装していることが多い。

3.3.1　キャッシュメモリ

- 以下の議論は1次キャッシュについて述べている。2次キャッシュは値は変わるが議論は本質的に同じ
- キャッシュは何故有効か？　→　アクセスの局所性
  時間的局所性：最近アクセスされたワードが再度アクセス
  空間的局所性：最近アクセスされたワードの近くがアクセスされる確率が高い



3.3.1　キャッシュメモリ

- ヒット率：アクセスする命令やデータが、キャッシュ・メモリにある確率。いかに少ないキャッシュでヒット率を向上させるかが問題。
- 平均メモリ・アクセス時間＝キャッシュアクセス時間×ヒット率＋メモリアクセス時間×（1－ヒット率）
- ブロックの大きさ：実験的結果から16から64バイトが採用されている（図3．23参照）。

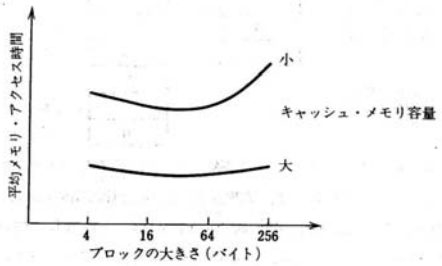
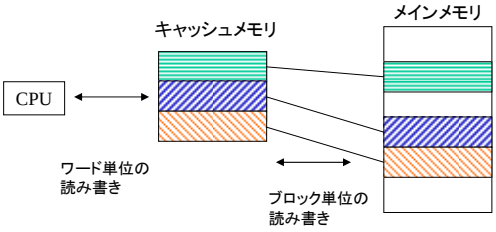


図 3.23　ブロックの大きさと平均メモリ・アクセス時間



3.3.1　キャッシュメモリ

マッピング機構

- ディレクトリ：キャッシュメモリと主記憶の対応関係を記憶。エントリは、空き/有りかの1ビットのフラッグと対応する主記憶のブロック番号ブロック番号（主記憶アドレスの上位nビット）
- 番号ブロック番号からキャッシュメモリのアドレス（エントリー番号）をみつける高速検索機能が必要となる。

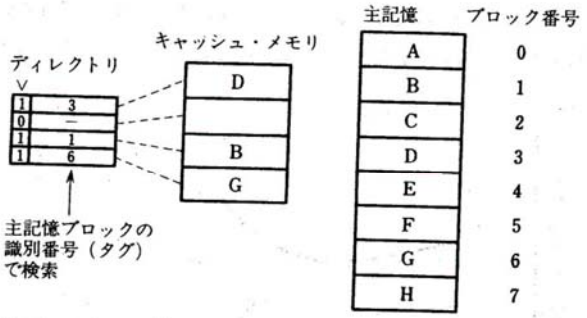


図 3.24　キャッシュ・ディレクトリの構成（この例ではタグ＝ブロック番号）

3.3.1　キャッシュメモリ

マッピング機構（ディレクトリ）の考慮点：大容量で高速の連想記憶実現が困難。
- 連想記憶：内容からアドレスを出力するメモリ。

- これを解決するために各種の方式が提案されている。
- 以下の例（一般的な妥当な値）
  - ブロックの大きさ：64バイト
  - キャッシュメモリ容量：64Kバイト(2^{16}バイト) →1Kブロック
  - 主記憶の最大容量：4Gバイト(2^{32}バイト)

主記憶アドレス（32ビット）

ブロック番号（26ビット）	6ビット

ブロック内アドレス

キャッシュメモリ説明例

- ブロックの大きさ：64バイト→2バイト
- キャッシュメモリ容量：64K(2^{16}) →8バイト(2^3バイト)
- 主記憶の最大容量：4G(2^{32}) → 16バイト(2^4バイト)

説明の都合で作った非常に小さな例

主記憶アドレス（4ビット）　　ブロック内アドレス

ブロック番号（３ビット）	1ビット

キャッシュメモリ

000	C1	C2	001
010	C15	C16	011
100			101
110	C5	C6	111

主記憶

0000	C1	C2	0001
0010	C3	C4	0011
0100	C5	C6	0101
0110	C7	C8	0111
1000	C9	C10	1001
1010	C11	C12	1011
1100	C13	C14	1101
1110	C15	C16	1111

キャッシュメモリ説明例

- プログラムは主記憶アドレスしかしらない
- 主記憶アドレスからキャッシュアドレスへの変換が必要
  →ディレクトリ（連想記憶：入力が内容で出力が番地）

C5の主記憶アドレス0100→ｷｬｯｼｭｱﾄﾞﾚｽ *11*0に変換が必要
C12のアドレス1011　　→ｷｬｯｼｭﾒﾓﾘにないことを検出が必要
赤字の部分がブロック番号

ディレクトリ

00	1000
01	1111
10	0000
11	1010

キャッシュメモリ

000	C1	C2	*001*
010	C15	C16	*011*
100			*101*
110	C5	C6	*111*

主記憶

0000	C1	C2	0001
0010	C3	C4	0011
0100	C5	C6	0101
0110	C7	C8	0111
1000	C9	C10	1001
1010	C11	C12	1011
1100	C13	C14	1101
1110	C15	C16	1111

3.3.1　キャッシュメモリ（完全連想方式）

- 完全連想方式
  - 26ビット×1024行の高速連想記憶が必要→実現困難

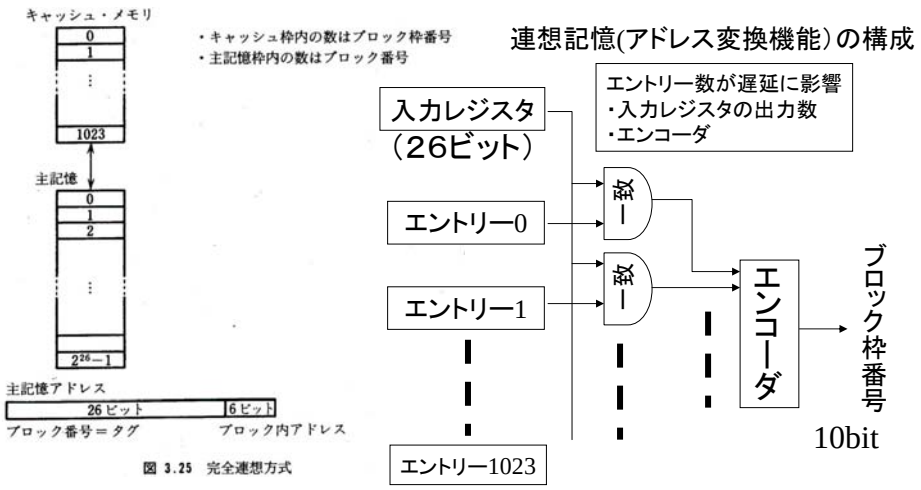
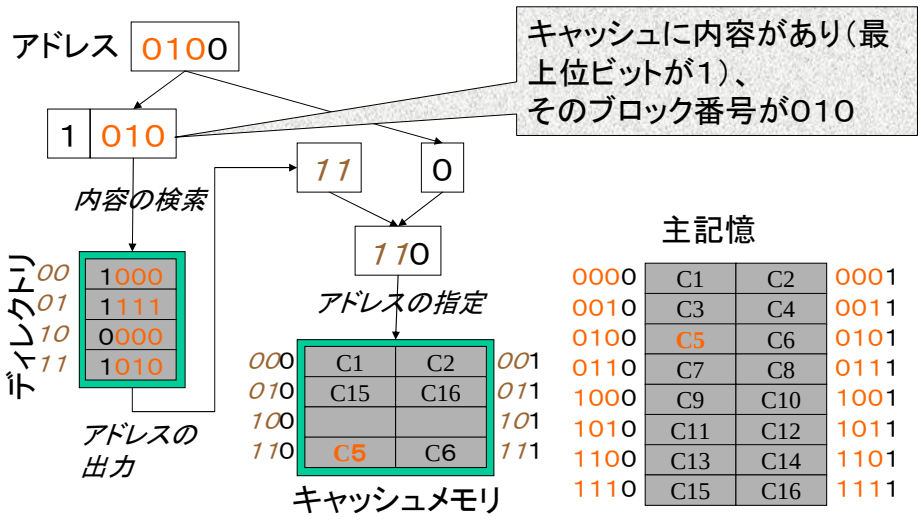


図 3.25　完全連想方式

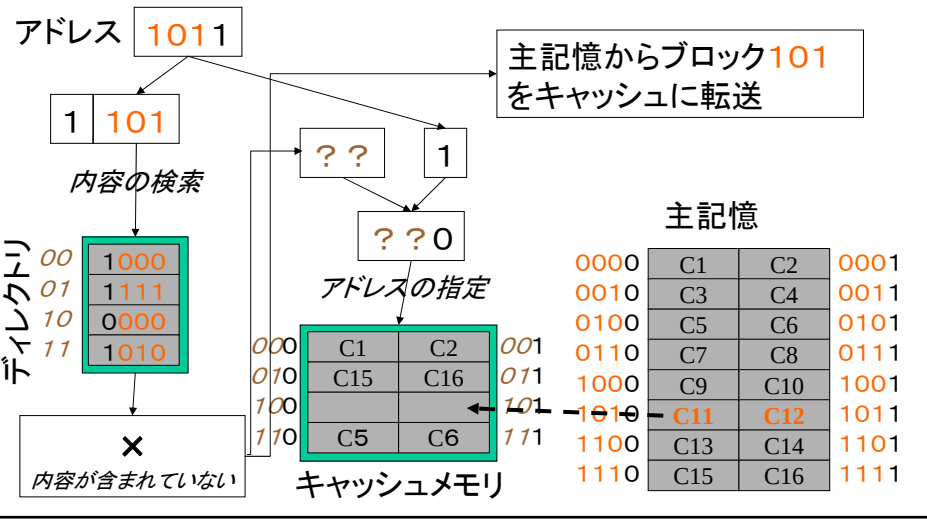
完全連想方式動作１説明例

例：C5の内容を読む（プログラム上でのアドレスは0100）



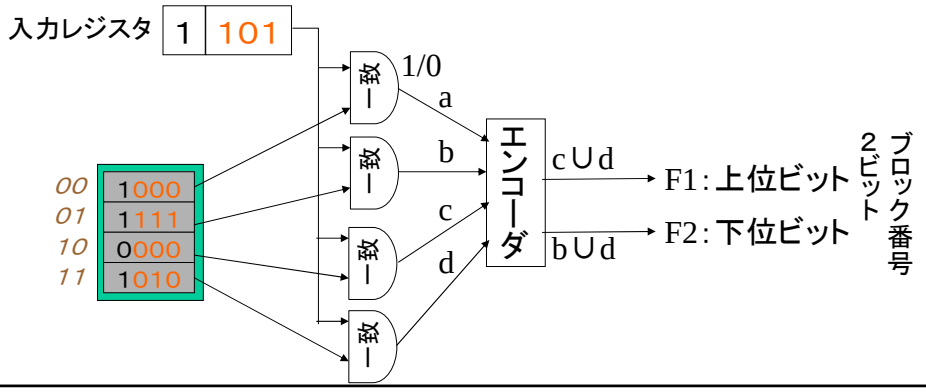
完全連想方式動作２説明例

例：C12の内容を読む（プログラム上でのアドレスは1011）



完全連想方式ディレクトリ説明例

- ディレクトリ(連想記憶)の機能:入力レジスタと内容が一致した時に番地を出力
- エントリー数(ブロック数)が多いと遅延が大きくなり実現困難



3.3.1　キャッシュメモリ（直接マップ方式）

- 直接マップ方式(図3．26):連想記憶16ビット×1行が1024個
- 制限条件:ブロック番号i≡j mod1024となるブロックiとjは同時にキャッシュできない。

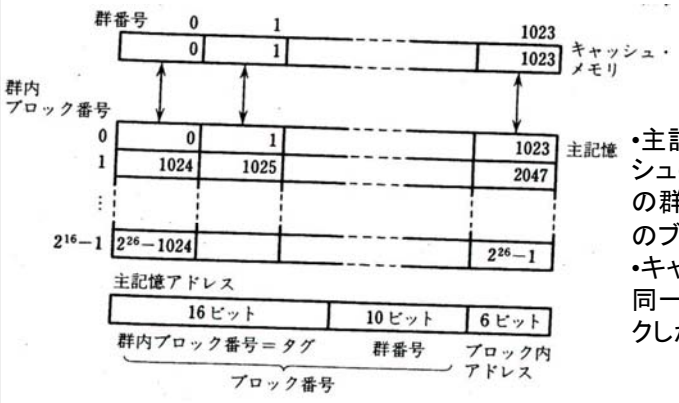
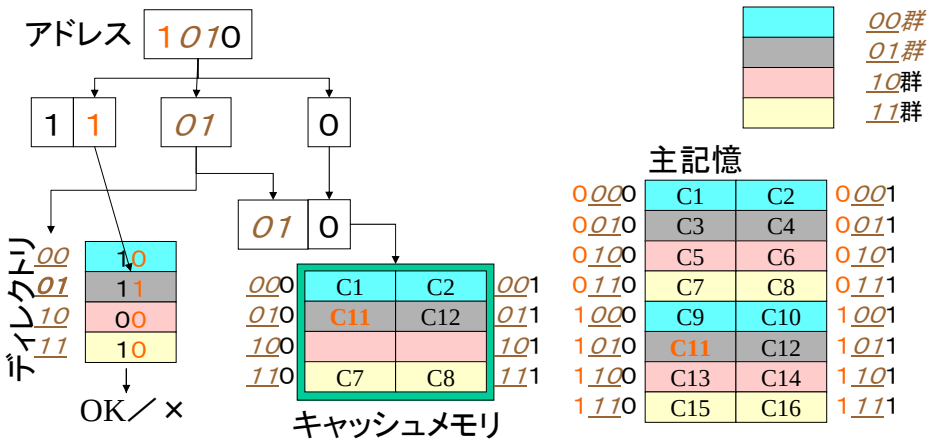


図 3.26　直接マップ方式

- 主記憶のブロックをキャッシュの枠数(1024)と同数の群に分割(各群は2^{16}個のブロック)
- キャッシュのブロックには同一の群番号をもつブロックしかおかない。

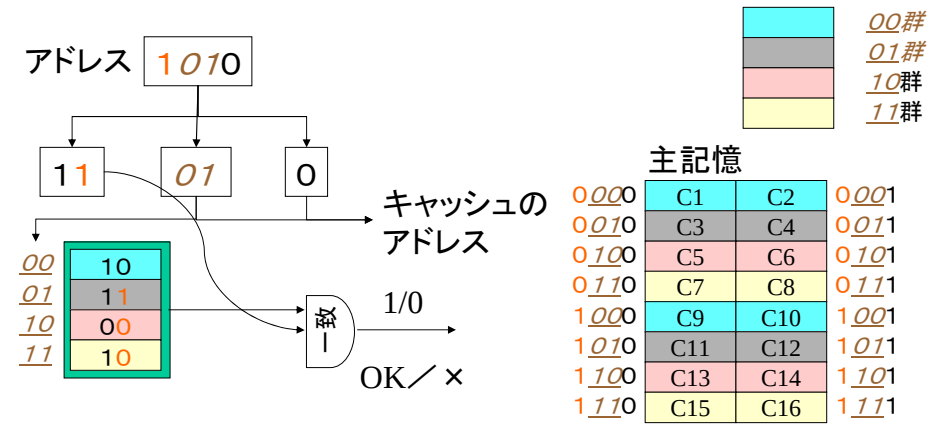
直接マップ方式動作説明例

- 主記憶のブロックをキャッシュのブロック数(4)と同数の群に分割(各群は2個のブロック)
- キャッシュのブロックには同一の群番号をもつブロックしかおかない。



直接マップ方式ディレクトリ説明例

- キャッシュのブロック数を同数のメモリと一つの一致回路で実現可能



3.3.1直接マップ方式

- 同一群で一つのブロックしかキャッシュにおけないのでヒット率が極端に低下する可能性がある。

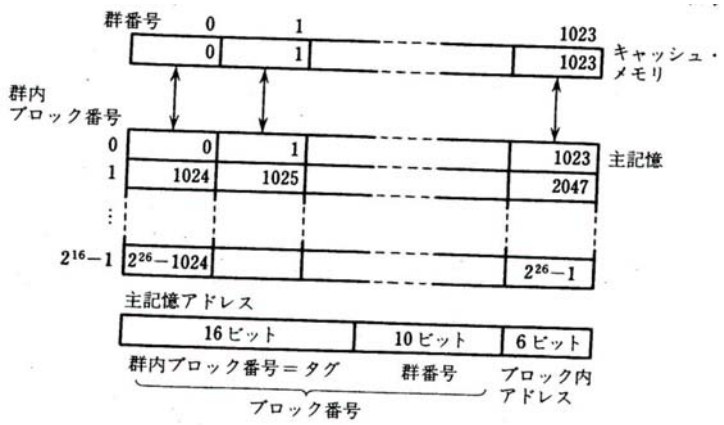


図 3.26 直接マップ方式

3.3.1 キャッシュメモリ

- 群連想方式（図3．27）：連想記憶20ビット×16行×64個
- 制限条件：ブロック番号i≡j mod63となるブロックiとjは同時に16個しかキャッシュできない。

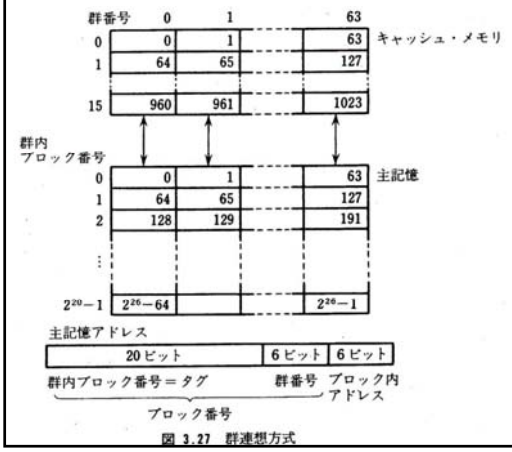
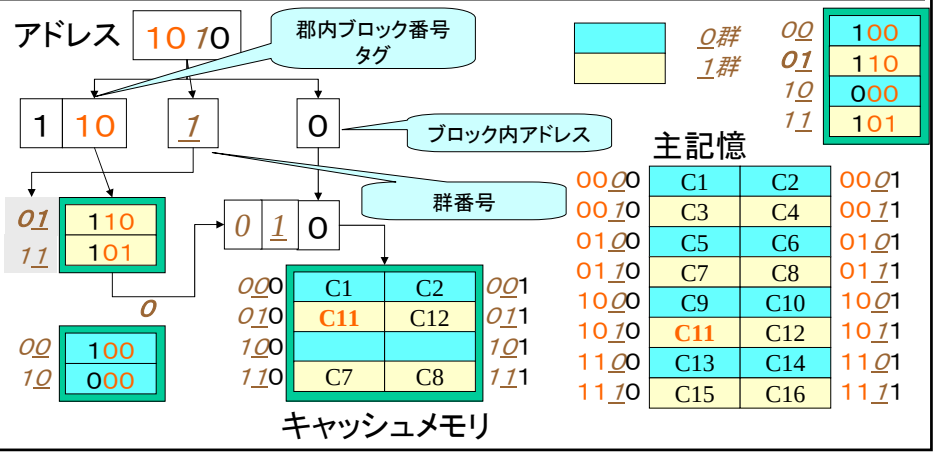


図 3.27 群連想方式

•主記憶のブロックをキャッシュの枠数（1024）÷16と同数の群に分割
•キャッシュも1024÷16の群に分割（一つの群に16個のブロックを割り当てる)
•キャッシュのブロックには同一の群番号をもつブロックしかおかない。

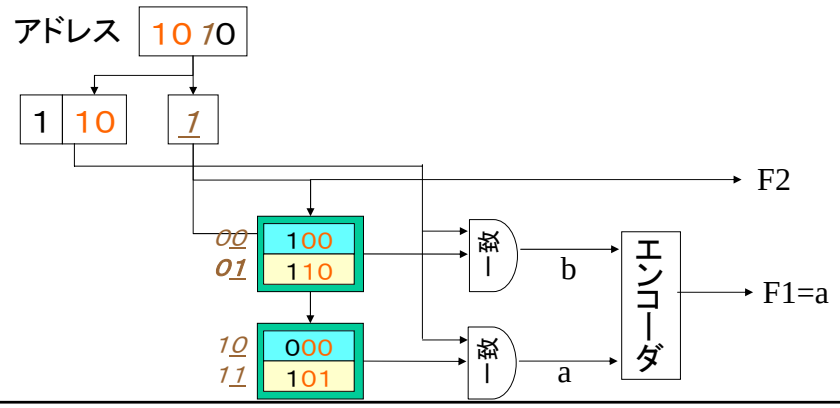
群連想方式動作説明例

- 主記憶のブロックをキャッシュのブロック数(4)より小さい数の群(2)に分割
- キャッシュのブロックには同一の群番号をもつブロックしかおかない。



群連想方式ディレクトリ説明例

- 一つの連想記憶のエントリ数
  ＝キャッシュのブロック数÷群数
- 下記の例は、理解容易にため。実際には連想記憶を群の数だけ容易する。



3.3.1　キャッシュメモリ

ブロックの大きさ（キャッシュメモリ容量が一定）

ブロックサイズ	小さい	→	大きい
ヒット率	小さい	→大きい←	小さい
キャッシュアクセス時間	大きい	→	小さい
ブロック転送の効率	悪い	→	良い

ブロック転送効率：メモリアクセス時間／1ブロックのワード数

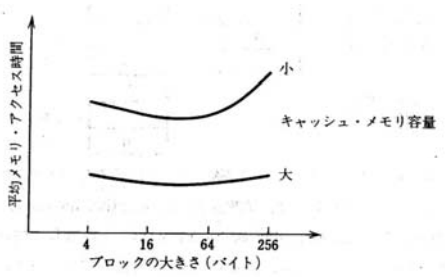
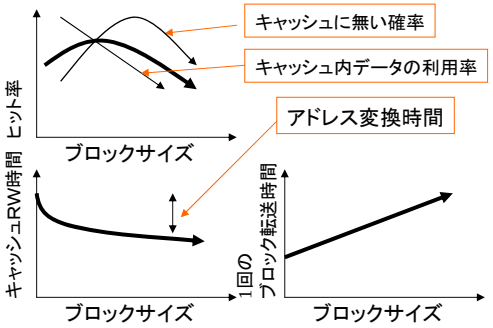


図 3.23　ブロックの大きさと平均メモリ・アクセス時間



(2)　読み出しと書き込み

- 読み出しの高速化（群連想方式）：ディレクトリの検索とキャッシュの読み出し（図3．24）を並列化（キャッシュにデータが入っているとすると、可能性は同一群内の16箇所しかない。）

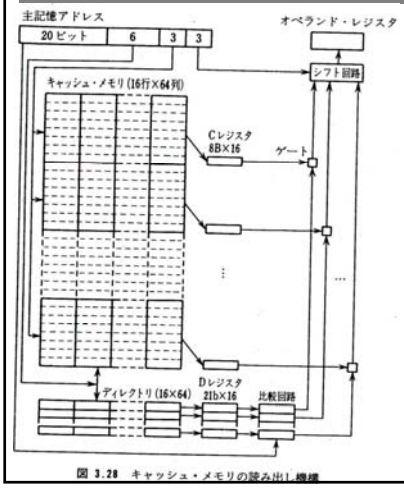


図 3.28　キャッシュ・メモリの読み出し機構

(2)　読み出しと書き込み

書き込み（ストアスルーとストアーイン方式）

- ストアスルー（ライトスルーとも言う）：プロセッサがデータをメモリに書き込む場合、キャッシュと同時にメインメモリにも書き込む方式
  - 書き込みの時間はメインメモリのアクセス時間と同じなので、高速化はされない。
  - しかしキャッシュの内容をメモリに追い出す必要が生じても、何もしなくてもよいので、回路が簡単になる。
- ストアイン（ライトバックとも言う）方式：書き込みはキャッシュメモリに行う。
  - 読み出し時間だけではなく、書き込み時間をも短縮している。
  - しかし書き込まれたデータはキャッシュメモリ上にしか存在しないため、キャッシュの内容をメモリに追い出すときは、キャッシュメモリの内容をメインメモリに書き戻さなければならない。
- 後者の方式は、前者の方式よりも実装は困難だが、全体的な性能はよくなると言われている。

(3) 置き換えアルゴリズム

キャッシュにデータがなく、しかもどのキャッシュにもメモリの写しが入っている時、どのブロックを追い出すか？

- どのやり方でもヒット率にほとんど差がない。

方法１：ランダムに選択
方法２：LRU法 (Least Recently Used)
など

(4)キャッシュメモリの多段構成

メインメモリに速度に差のあるDRAMとSRAMを用いるために、2次キャッシュを用いる（表３．６参照）。

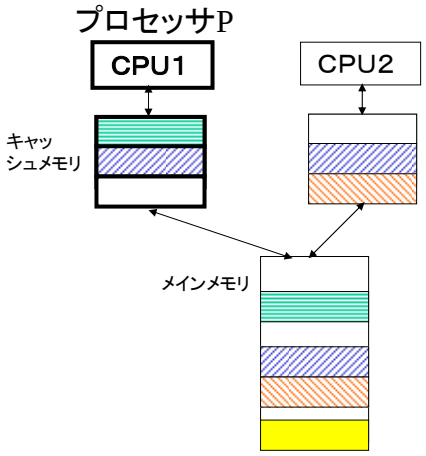
- DRAM(Dynamic RAM):bitの情報を記憶するメモリセルが、コンデンサとトランジスタ1つずつで構成されているRAM。コンデンサに電荷が溜まっているかどうかで“1”か“0”かを判別する。このDRAM内部のコンデンサは、放置しておくと自然に放電してデータを失ってしまうという特性がある。そのため、完全に放電してしまう前にコンデンサを再充電する必要がある。これをリフレッシュという。DRAMには必ず一定期間内にリフレッシュサイクルを必要なだけ与えなければならない。リフレッシュサイクル中はデータの読み書きができず、CPUからのアクセスも待たされるため、速度低下の一因となる。このような理由もあって、DRAMはSRAMより遅くなる傾向がある。しかし記録密度については、同程度の製造技術を用いた場合、DRAMはSRAMの約4倍の密度を実現できる。
- SRAM(Static RAM)：4～6個のトランジスタで構成されたメモリセルでbitの情報を記憶するRAM。DRAMに比べると、コンデンサに充電しないため、アクセスタイムを大幅に高速化できる

3.3.1　キャッシュメモリ（再掲）

- 一般的に、メインメモリに使われているDRAMのスピードはCPUに比べてかなり遅く、CPUの命令実行速度を下げる原因となっている。
- この問題を解決するために、CPUとメインメモリの間にキャッシュメモリと呼ばれる高速／小容量のメモリを配置する。
- メインメモリからキャッシュメモリへはブロック（数語から数十語単位）にまとめて転送する手法がとられる。
- メモリシステムの高速化のために、キャッシュメモリが2段、3段と重ねて実装されることがある。この場合、CPUに近い位置にあるほうから1次キャッシュ、2次キャッシュ…と呼ばれる。
- 486以降のx86 CPUは1K～16Kbytes程度の1次キャッシュをCPU内部に内蔵している。
- また現在のPC互換機では、高速SRAMを用いて64K～1Mbytes程度の2次キャッシュを実装していることが多い。

（５）キャッシュメモリの一致制御

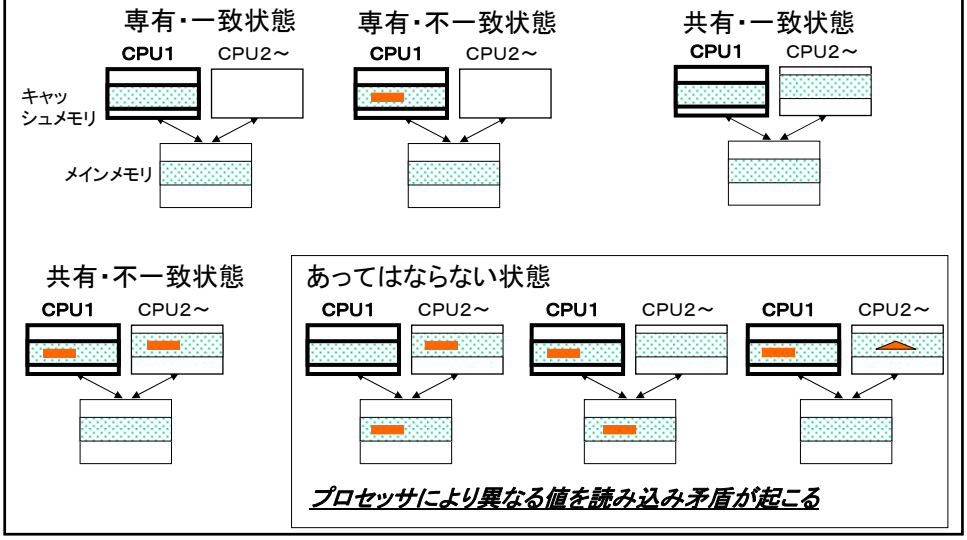
- 2台以上のキャッシュをもつプロセッサが同一メモリの同じブロックにアクセスする。



あるブロックの（あるプロセッサPから見た）状態
- 無効状態：Pのキャッシュにない
- 有効・専有状態：Pのキャッシュにあって、内容がメインメモリと一致し、他のプロセッサのキャッシュに存在しない。
- 有効・共有状態：Pのキャッシュにあって、内容がメインメモリと一致し、他のプロセッサのキャッシュにも存在する。

（５）キャッシュメモリの一致制御

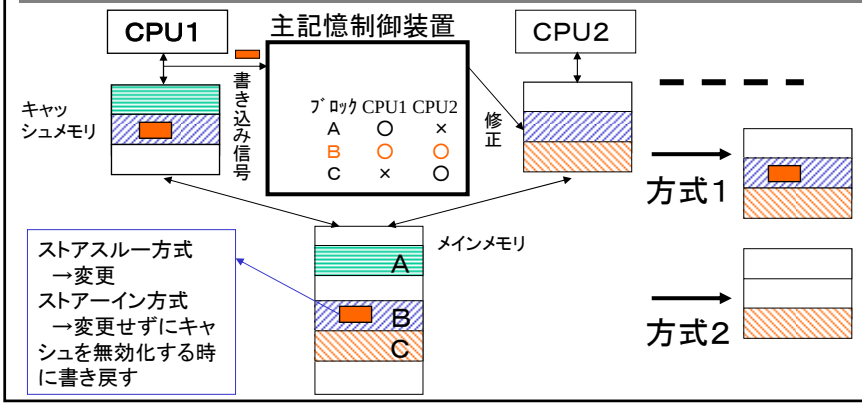
- 不一致状態が起こる



（５）キャッシュメモリの一致制御

集中制御方式

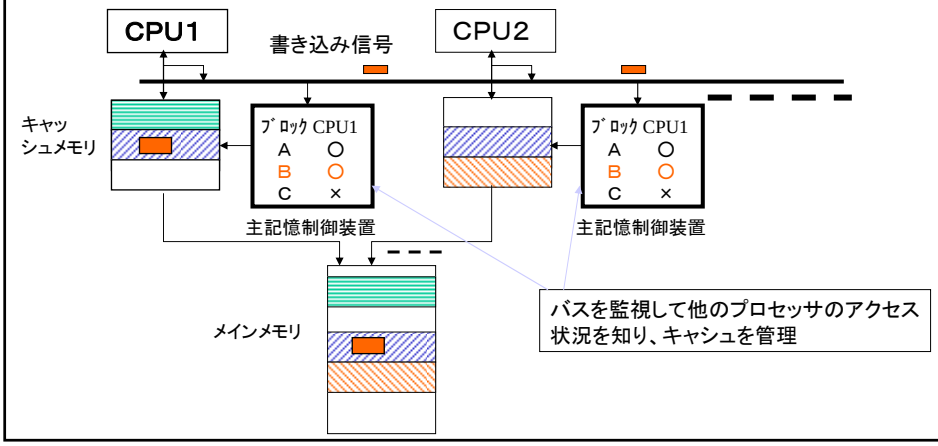
- 主記憶制御装置がどのプロセッサが写しをもっているか管理
- 書き込み信号をうけると該当のブロックをもつ他のキャッシュを修正
  - 方式１：値を修正
  - 方式２：キャッシュを無効化



（５）キャッシュメモリの一致制御

分散制御方式（各プロセッサが独立に管理）

- 書き込み信号を他プロセッサに流す(読み出しは独立に行える)。
- 各プロセッサのキャシュ管理装置は、他プロセッサの書き込み信号を監視してキャッシュを管理

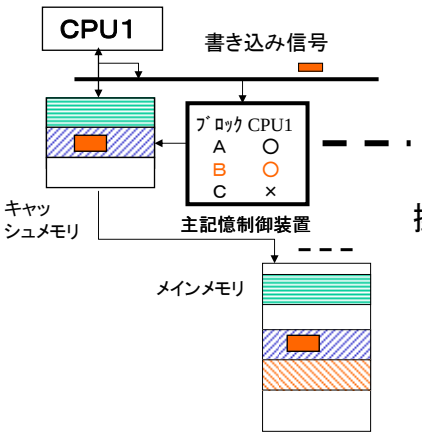


（５）キャッシュメモリの一致制御

分散制御方式（各プロセッサが独立に管理）

- 状態と操作の組み合わせを考えで、システムを設計する。

（並行に動作するシステムでよく用いられる手法であるので、手法が大切。）



状態

- 無効状態
- 専有・一致状態
- 専有・不一致状態
- 共有・一致状態
- 共有・不一致状態

操作

自プロセッサ	他プロセッサ
ＲＨ	ｒｈ
ＲＭ	ｒｍ
ＷＨ	ｗｈ
ＷＭ	ｗｍ

W,ｗ：書き込み　R,ｒ：読み込み
H,ｈ：キャシュ　M,ｍ：メインメモリ

（５）キャッシュメモリの一致制御

更新型書き込み一致制御方式

- 主記憶にはストアスルー方式
- 無効状態と有効・一致・（専有か共有）状態しかないようにシステムを制御する。
- 専有と共有では動作は同じなので教科書の表3.7では１状態で表現している。

ブロックの状態	自キャッシュの動作				他キャッシュの動作			
	ＷＨ	ＲＨ	ＲＭ	ＷＭ	ｗｈ	ｒｈ	ｒｍ	ｗｍ
VAL1 有効・一致・共有	VAL1 $C_A \leftarrow P_A$ $C_B, M \leftarrow C_A$	VAL1 *NOP*			VAL1 $C_A \leftarrow C_B$	VAL1 *NOP*	VAL1 *NOP*	VAL1 $C_A \leftarrow C_B$
VAL2 有効・一致・専有	VAL2 $C_A \leftarrow P_A$ $M \leftarrow C_A$	VAL2 *NOP*					VAL1 *NOP*	VAL1 $C_A \leftarrow C_B$
INV 無効			VAL1/2 $C_A \leftarrow M$ $P_A \leftarrow C_A$	VAL1/2 $C_A \leftarrow M$ $C_A \leftarrow P_A$ $C_B, M \leftarrow C_A$	INV *NOP*	INV *NOP*	INV *NOP*	INV *NOP*

（５）キャッシュメモリの一致制御

更新型書き込み一致制御方式の状態遷移図

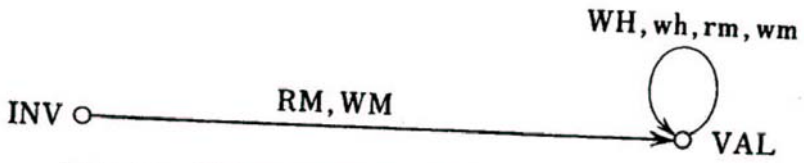


図 3.29　更新型書き込み一致制御方式の状態遷移図

主記憶へすべてのプロセッサから書き込み（ライトスルー方式）が前提であり、主記憶書き込みがボトルネックになる可能性がある。

（５）キャッシュメモリの一致制御

無効型書き込み一致制御

- 有効・不一致・専有状態もゆるす（ストアイン）
- 有効・一致・共有状態で書き込み
  →他のプロセッサのキャッシュから該当ブロックを消去（無効状態に）して有効・不一致・専有状態にする。
- 有効・不一致・専有状態で、他のプロセッサに読み込み
  →有効・一致・共有状態にする。

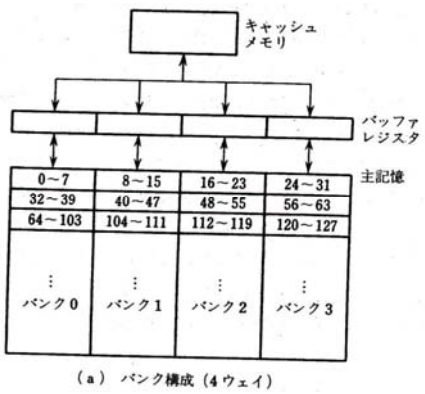
（５）キャッシュメモリの一致制御

ブロックの状態	自キャッシュの動作				他キャッシュの動作			
	WH	RH	RM	WM	wh	rh	rm	wm
VAL1 有効・一致 ・共有	DIR $C_A \leftarrow P_A$ wh送出	VAL1 *NOP*			INV *NOP*	VAL1 *NOP*	VAL1 *NOP*	INV *NOP*
VAL2 有効・一致 ・専有	DIR $C_A \leftarrow P_A$	VAL2 *NOP*					VAL1 *NOP*	INV *NOP*
DIR 有効・不一致 ・専有	DIR $C_A \leftarrow P_A$	DIR *NOP*					VAL1 $C_B, M \leftarrow C_A$	INV $C_B, M \leftarrow C_A$
INV 無効			VAL1/2 $C_A \leftarrow C_B/M$ $P_A \leftarrow C_A$	VAL1/2 $C_A \leftarrow C_B/M$ $C_A \leftarrow P_A$ wm送出	INV *NOP*	INV *NOP*	INV *NOP*	INV *NOP*

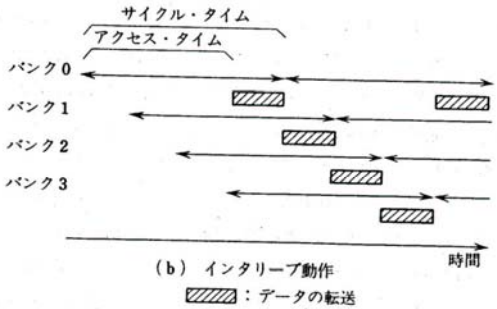
- 1つのキャシュに連続して書き込みがあった場合、1回目以外はバスにwhを送出する必要がない。

3.3.2　主記憶の構成

- 主記憶からキャッシュメモリへのブロック転送を並列化することにより、主記憶の読み出し時間を削減する手法
  - ブロック転送時間＝MRT＋CRT×ブロックワード数
    - MRT：主記憶からの読み出し時間(アクセスタイム)
    - CRT:キャッシュメモリへの書き込み時間(データ転送)



（a） バンク構成（4ウェイ）



（b） インタリーブ動作
：データの転送

図 3.31　主記憶のバンク構成とインタリーブ