計算機システムⅡ
キャッシュと仮想記憶

和田俊和

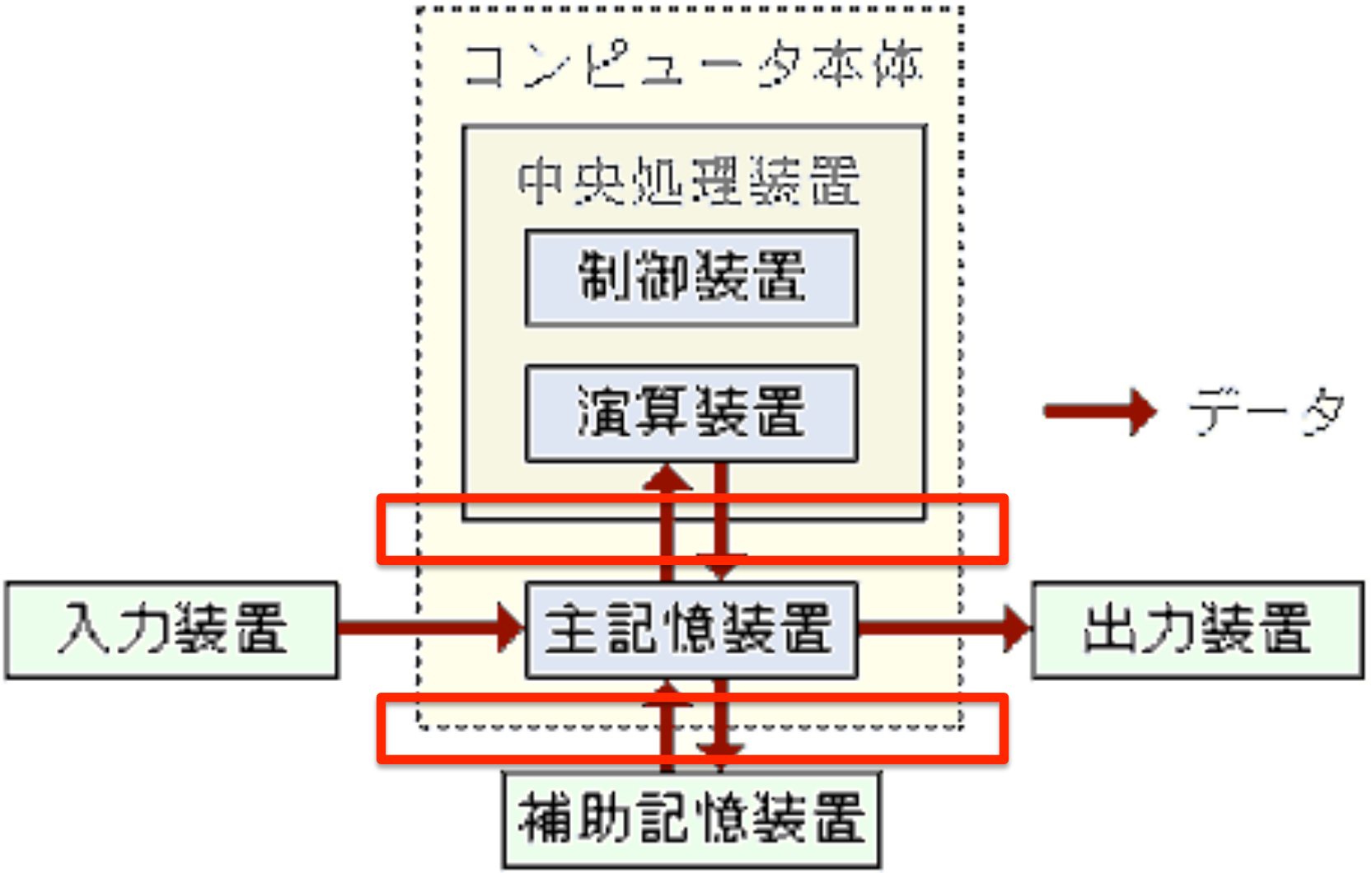
講義計画

1. コンピュータの歴史1
2. コンピュータの歴史2
3. コンピュータの歴史3
4. 論理回路と記憶，計算：レジスタとALU
5. 主記憶装置とALU，レジスタの制御
6. 命令セットアーキテクチャ
7. 演習問題
8. パイプライン処理
9. メモリ階層：キャッシュと仮想記憶（←本日）
10. 命令レベル並列処理
11. 命令実行順序の変更
12. 入出力と周辺装置：DMA,割り込み処理
13. 演習問題
14. 現代的な計算機アーキテクチャの解説
15. 総括と試験

- 教科書：坂井修一著：電子情報通信学会レクチャーシリーズC−9，コンピュータアーキテクチャ，コロナ社

- 最終回の試験によって成績評価を行う．5回以上欠席で不合格とする．

本日の講義の範囲



５．１　記憶階層（キャッシュと仮想記憶を包含する，総論）

5.1.1 命令パイプラインとメモリ

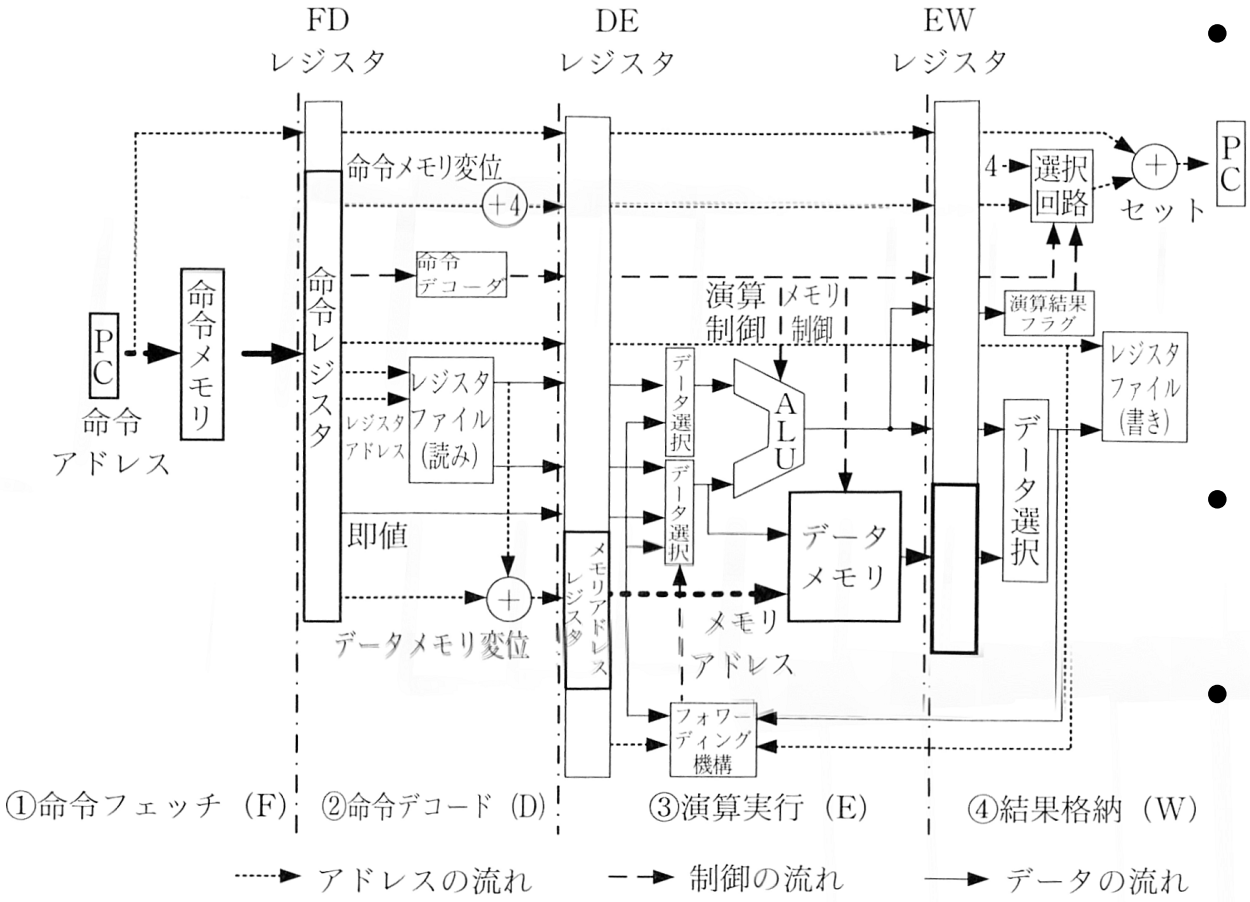
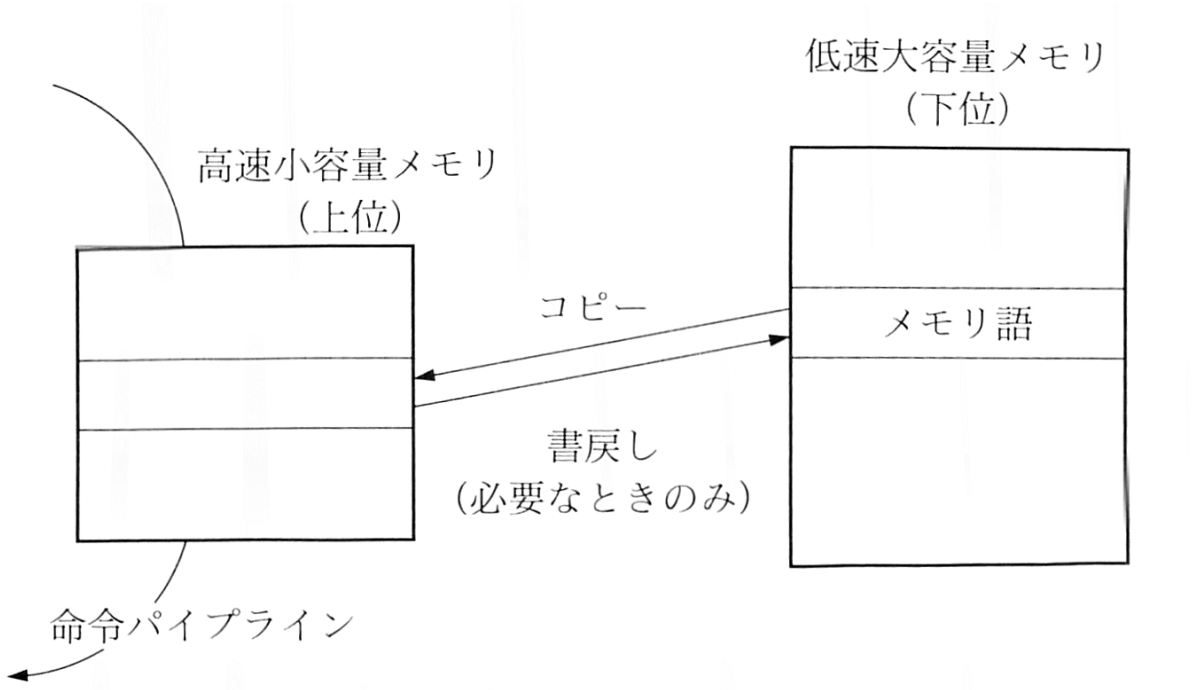


図 5.1 命令パイプライン（太線はメモリ操作）

- パイプラインを動かすためには，メモリの読み書きを1クロックで済ませる必要がある.
- バス駆動を伴うメモリアクセスは遅い.
- 遅いメモリのコピーを高速なメモリにとって使う

5.1.2記憶階層と局所性



1. 空間的局所性

   あるメモリ語が参照された際に，その周辺の語も参照され易い．

2. 時間的局所性

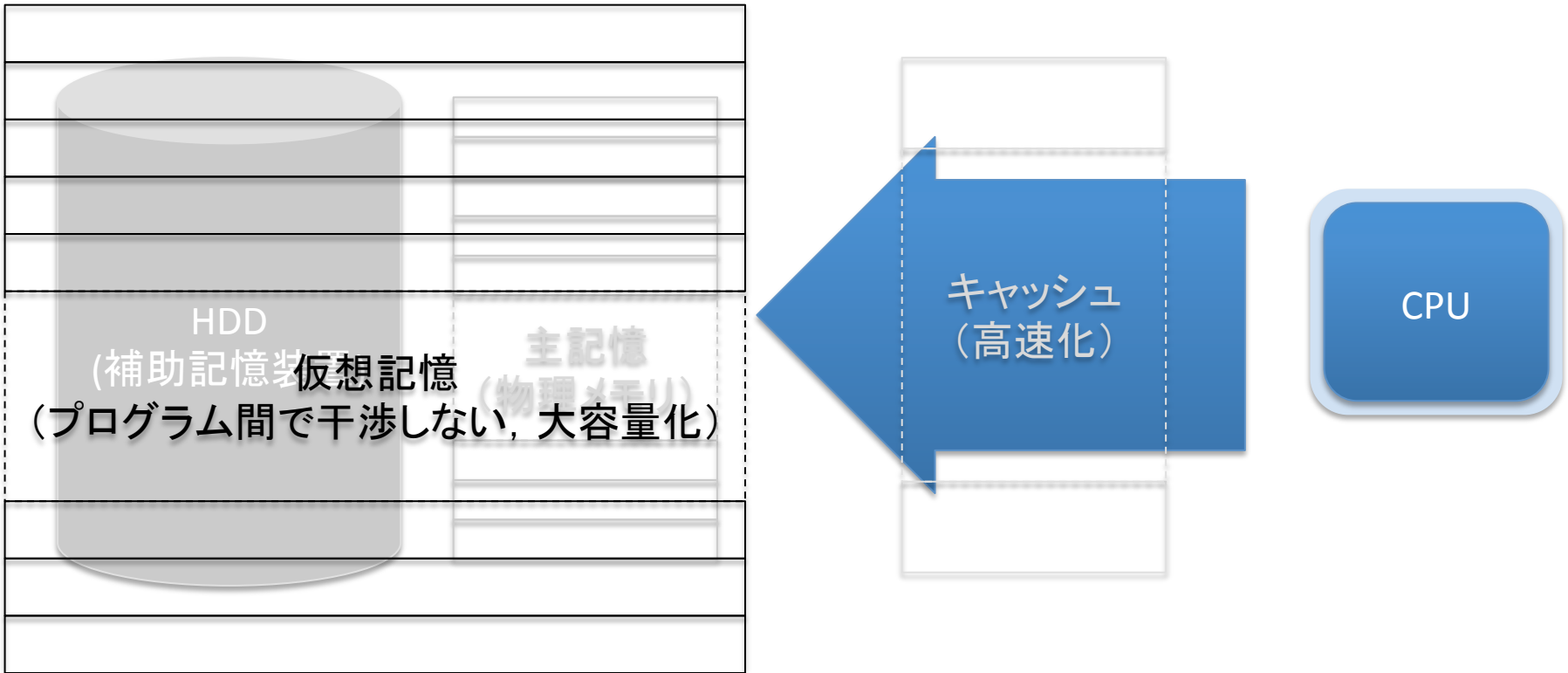
   あるメモリ語が参照された際に，その語が時間をおかずに再び参照され易い．

- 高速小容量メモリには，よく使われる命令やデータが格納される．
- 低速大容量メモリには，プログラムカウンタが指すことの出来る全ての命令と，ロード，ストアできる全ての命令が格納されている．

人間も同じ．長期記憶と短期記憶がある．

5.1.3 透過性

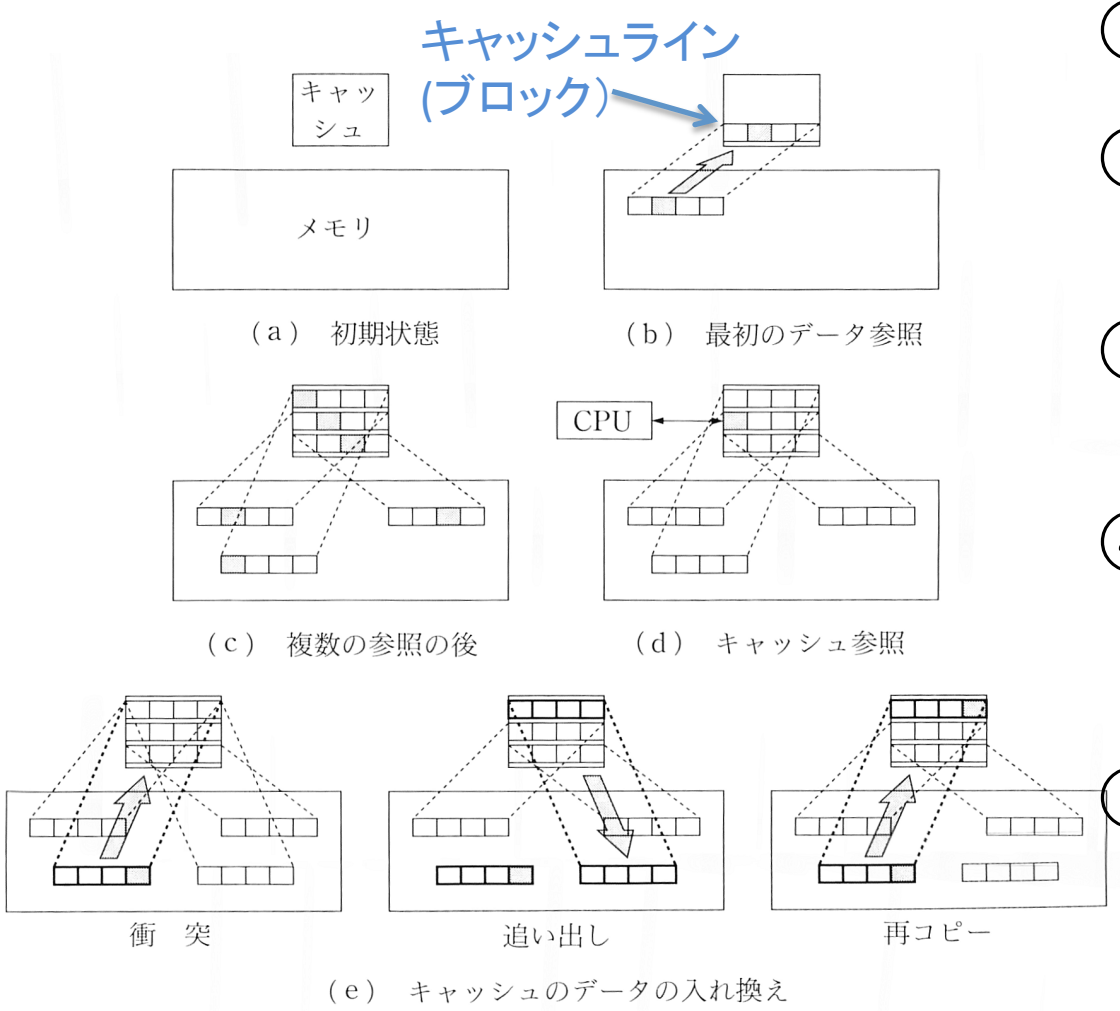
- 高速メモリへのデータのコピーや，メモリへの書き戻しを，プログラマに意識させない．
- CPUと主記憶の関係だけしか見えないようにする．



5.2 キャッシュ

5.2.1 キャッシュとは何か

- キャッシュは命令パイプラインの動作速度でデータの読み書きが出来なければならない.



①キャッシュには何も入っていない (a)
②最初のデータが参照されるとキャッシュにそのデータと周辺の数語のメモリも入れられる. (b)
③引き続きデータの参照が起きるとキャッシュにデータが入れられる. (c)
④メモリ参照時にはまずキャッシュが参照され，ここにデータがあれば，実際のメモリアクセスは生じない. (d)
⑤キャッシュがいっぱいになると不要なデータは捨てられ，新しいデータがキャッシュに入れられる. (e)

現金ではありません.

cache / kæʃ / (! cashと同音)

名詞

1 C (武器・食料・金などの)隠し[貯蔵]場所; 隠した物.

2 C U 《コンピュ》キャッシュメモリ(cache memory).

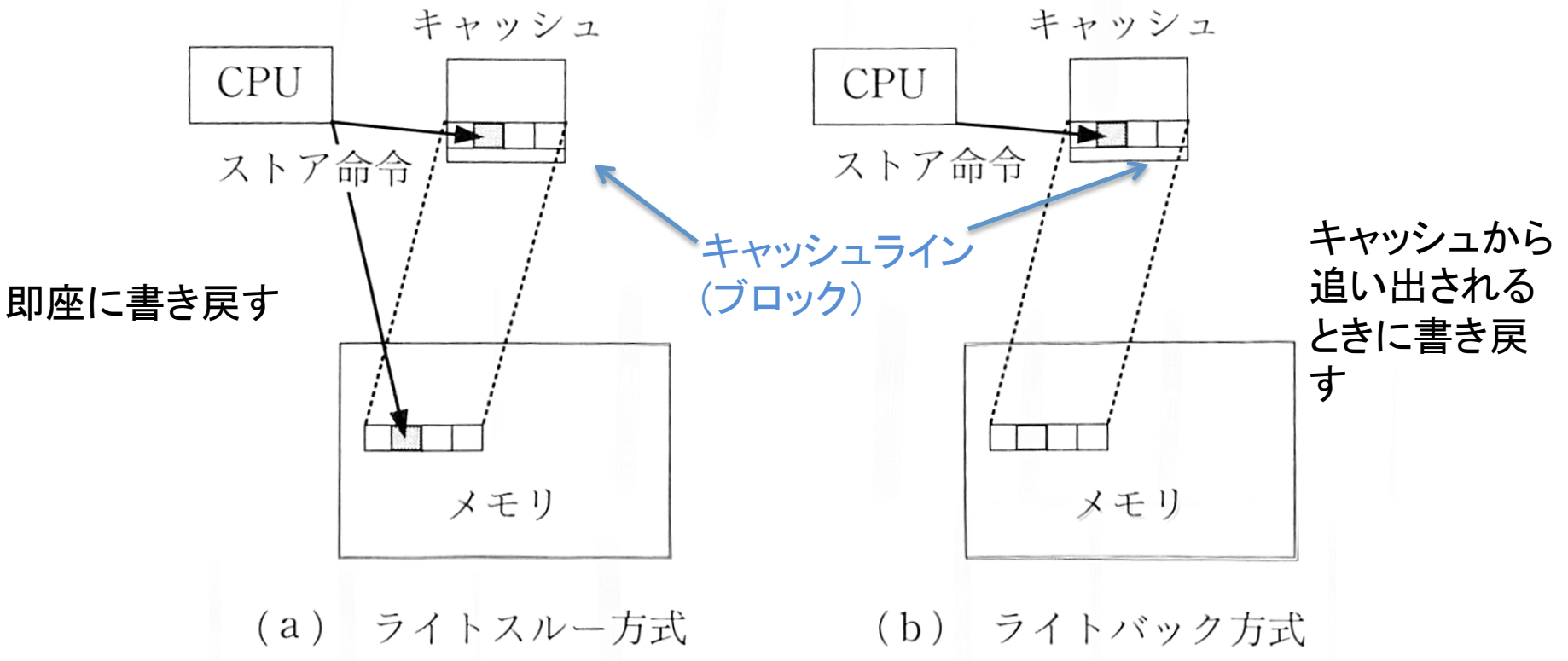
動詞

他動詞

…を隠す, 貯える; 《コンピュ》〈データ〉をキャッシュする
〖よく使うデータを一時的にメモリなどに記憶する〗.

5.2.2 ライトスルーとライトバック
Write through, Write back

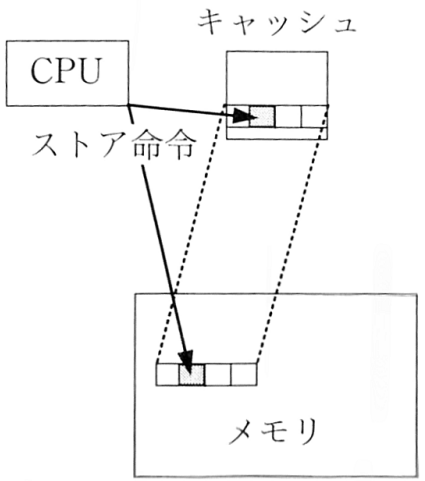
- CPUがキャッシュに対する書き込みを行った場合，元のメモリにもこの変更を書き戻す必要が生じる．このタイミングの違い．



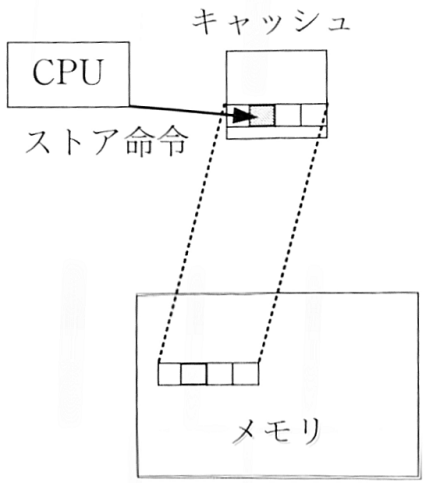
ライトバック，ライトスルーの比較

項目	Write through	Write back
メモリアクセス	ストア命令の実行時	キャッシュライン追い出しの時
Write命令の実行速度	Write bufferの速度	キャッシュの速度
キャッシュライン書き戻し	不要	キャッシュライン書き出しの時
実装	単純	複雑

ライトスルーの場合には速度が遅くなりすぎるので，キャッシュと，主記憶の間にwrite bufferという比較的高速なメモリを設けるのが普通である.



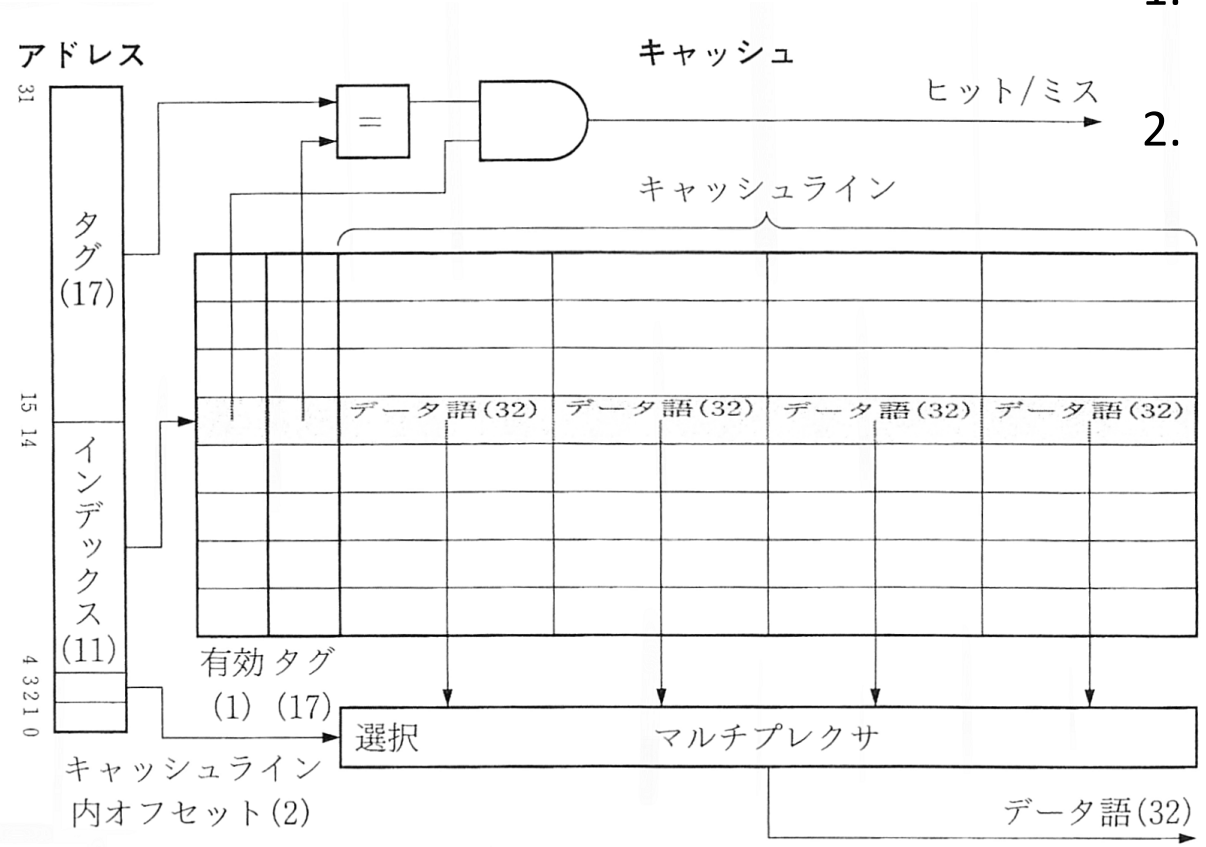
（a） ライトスルー方式



（b） ライトバック方式

5.2.3 ダイレクトマップ型キャッシュの機構と動作

- 読み出し:
  - タグ→求めるキャッシュラインかどうかの判定
  - インデックス→キャッシュライン上の位置



(a) 読出しの動作

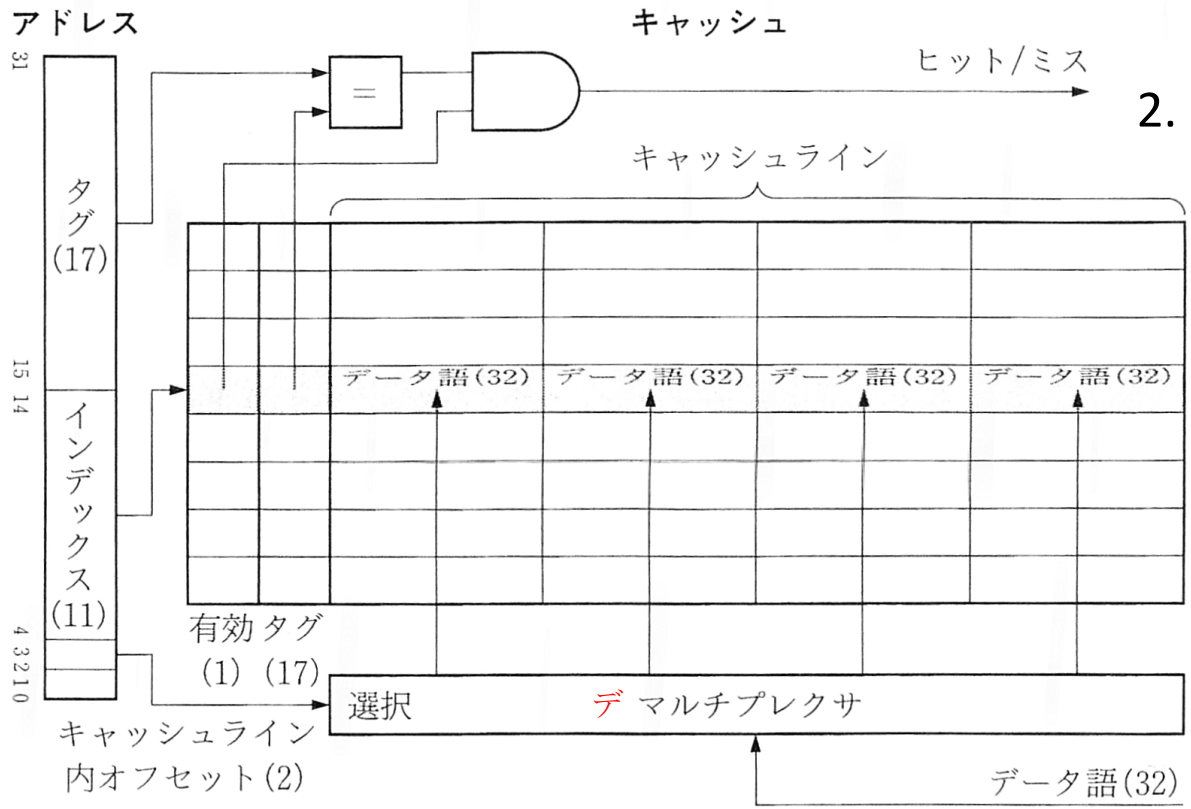
1. インデックスから，キャッシュラインとタグを読み出す．
2. メモリアドレスのタグと，タグを比較し，一致していればヒット，そうでなければミス

1. ヒットしていればキャッシュライン内オフセットを参照してキャッシュからデータを読み出す．
2. ミスしていた場合は，主記憶に書き戻し，メモリからここにデータを読み出す．そして，キャッシュからデータを読み出す．

ダイレクトマップ型キャッシュの機構と動作：

書き込み

- タグ→求めるキャッシュラインかどうかの判定
- インデックス→インデックス%ライン数＝キャッシュラインの番号



（b） 書込みの動作

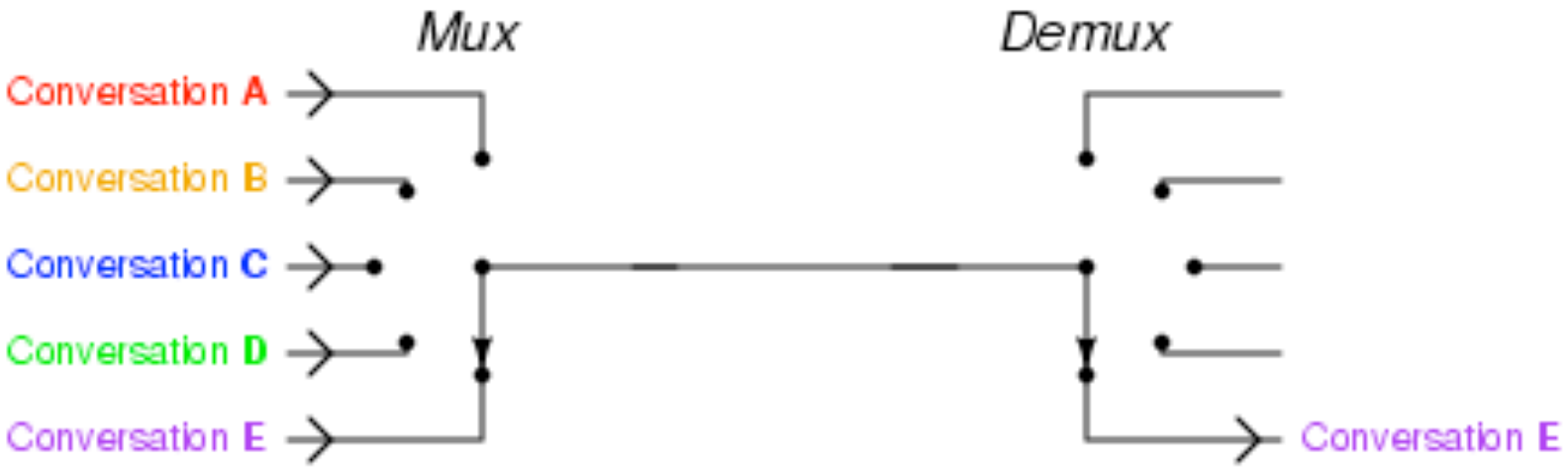
1. インデックスから，タグを読み出す．
2. 書き込みアドレスのタグと，1のタグを比較し，ヒットかミスかを判定

1. ヒットしていればキャッシュライン内オフセットを参照してキャッシュにデータを書き込む．

2. ミスしていた場合は，主記憶に書き戻し，ここに所望のデータを読み出してくる．その上で，オフセットを参照してキャッシュにデータを書き込む．

用語

- マルチプレクサ：多数の信号を一本のラインに乗せて送出するための機構
- デマルチプレクサ：一本のラインの信号を複数のラインにつなぎ替えて送出する機構.

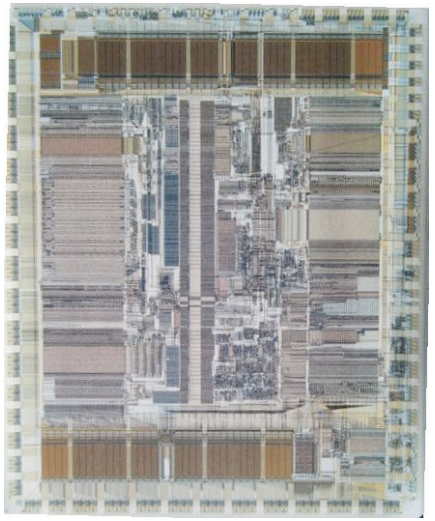


5.2.4 キャッシュミス

- 初期参照ミス: (compulsory miss, cold start miss)
  - 最初にキャッシュラインにアクセスすることで生じるミス
- 競合性ミス : (conflict miss, collision miss)
  - 同じインデックスを持つ異なるキャッシュラインにアクセスすることで生じるミス.
- 容量性ミス : (capacity miss)
  - キャッシュに入れたいラインの数がキャッシュの容量を上回ることで起こるミス.

3つのC

競合性ミスの実例：
Dec Alpha CPU 21064

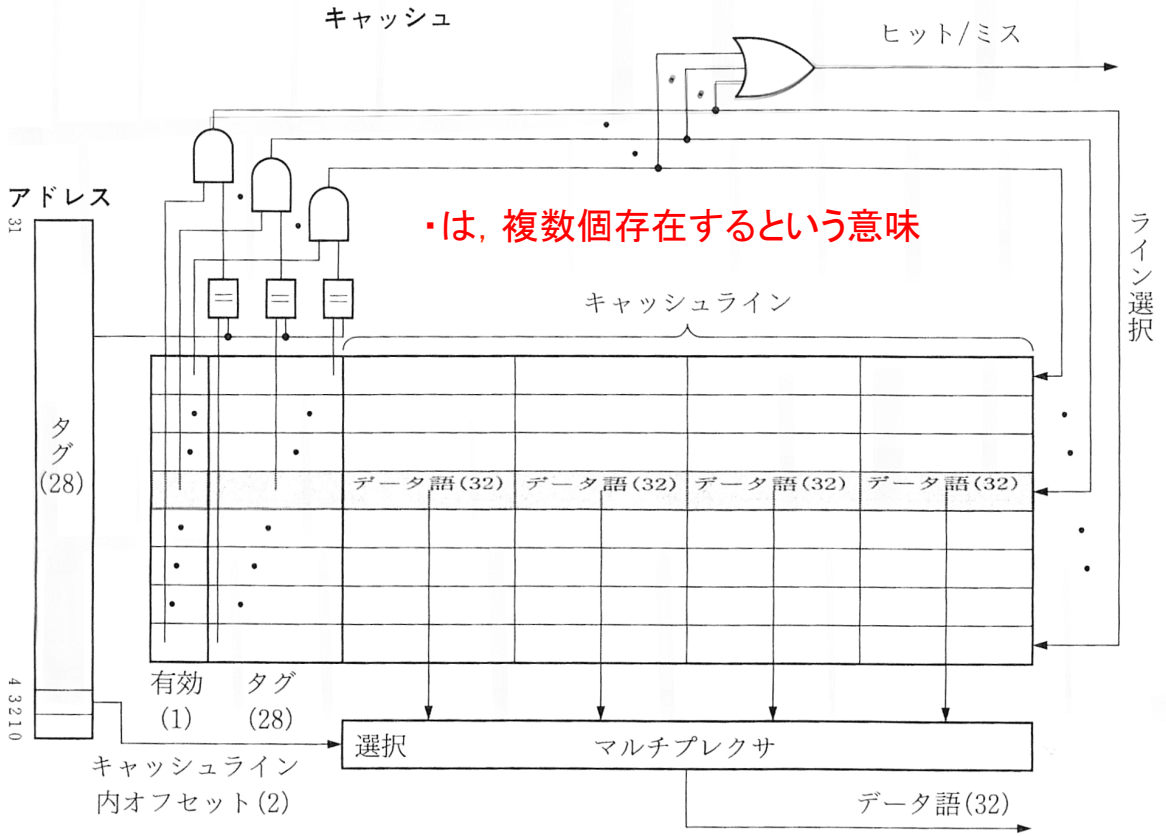


- 幅の広い銅配線でクロックを上げるだけという超高速CPU.
- キャッシュはダイレクトマッピング形のキャッシュのみ.
- 小技は殆ど使わない，王道の高速化路線.

- しかし，下記の背景差分計算をすると，何故か極めて速度が落ちた.
- ある単純なことをするだけで，全く同じアルゴリズムであるのに，速度が6倍も向上した.

5.2.5 フルアソシアティブ形キャッシュとセットアソシアティブ形キャッシュ

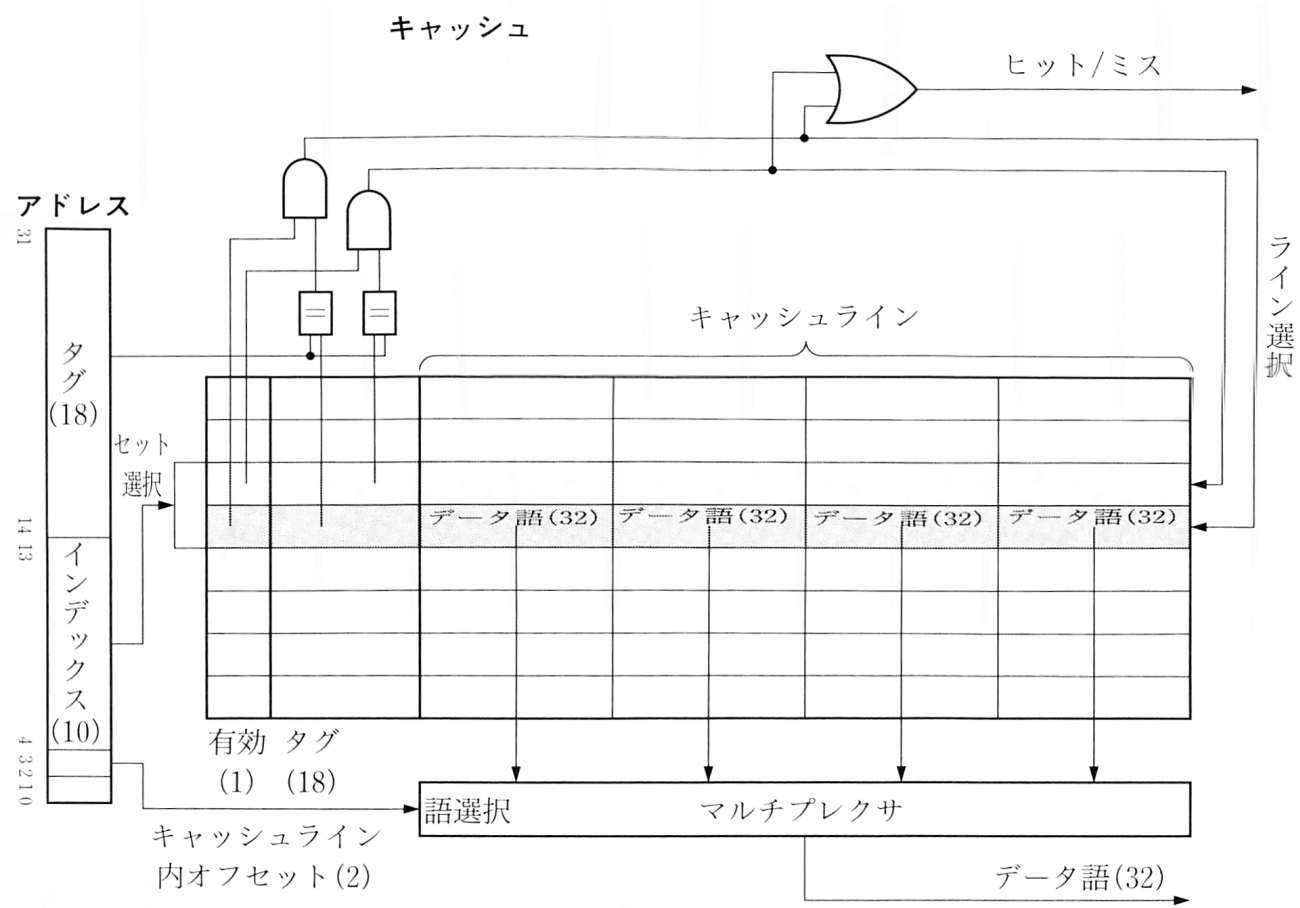
- ダイレクトマッピングは，高速であるが，競合性のミスが多発する可能性がある．
- インデックスを使わずタグだけでキャッシュラインを求める．



回路が大規模になり，遅延も発生しやすいため，小規模のキャッシュでしか用いられない．

セットアソシアティブ形キャッシュ

- インデックスの剰余によって決まるキャッシュラインを複数持つことで，キャッシュ競合を回避する.



一つのインデックスに対して，A本のキャッシュラインが保持される場合，Aを「連想度」と呼び，方式を「Aウエイのセットアソシアティブ形キャッシュ」と呼ぶ.

セットアソシアティブ形キャッシュ

- ライン数L，セット数S，連想度をAとすると

$$L = S \times A$$

フルアソシアティブは，S＝1とした場合．

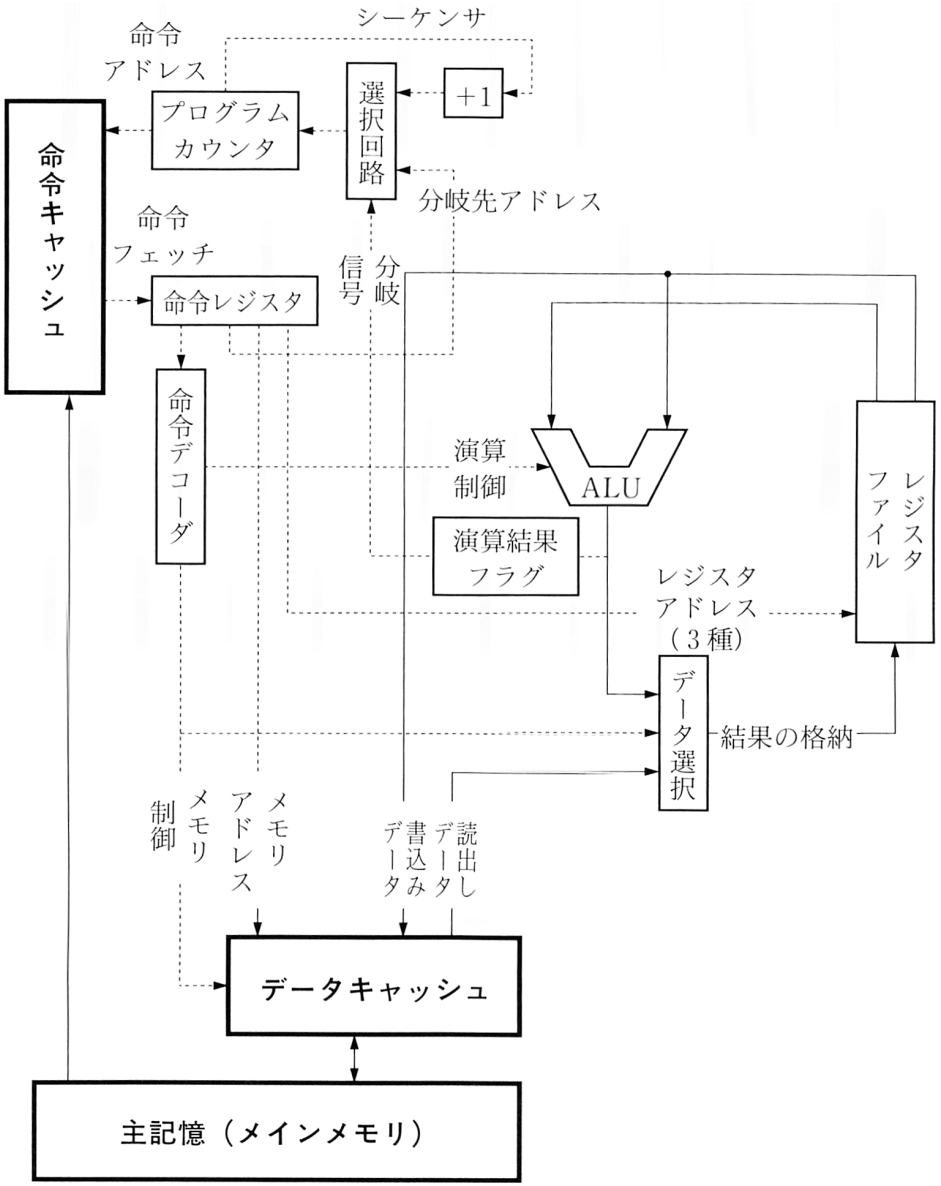
表5.2　キャッシュの三つの形

項　目	ダイレクトマップ	セットアソシアティブ	フルアソシアティブ
連想度	1	A（2，4など）	＝ライン数
セット数	＝ライン数	S	1
ハードウェア	◎	○	×
ゲート遅延	◎	○	×
競合性ミス	×	○	◎

質問はありませんか？

- ダイレクトマップ形キャッシュメモリの連想度Aはいくら？
- 何故，有効ビットがあるのでしょう？
- セットとは何でしょうか？

5.2.6 キャッシュの入ったCPU



- 命令キャッシュとデータキャッシュは通常分けておく.（パイプライン動作で競合が起きるのを避けるため）
- ミスの際は，パイプライン全体を止め，ラインをキャッシュからメモリに書き戻し，メモリから必要なラインをキャッシュに読み込んだ後，パイプラインの実行を行う.

キャッシュの入ったCPU

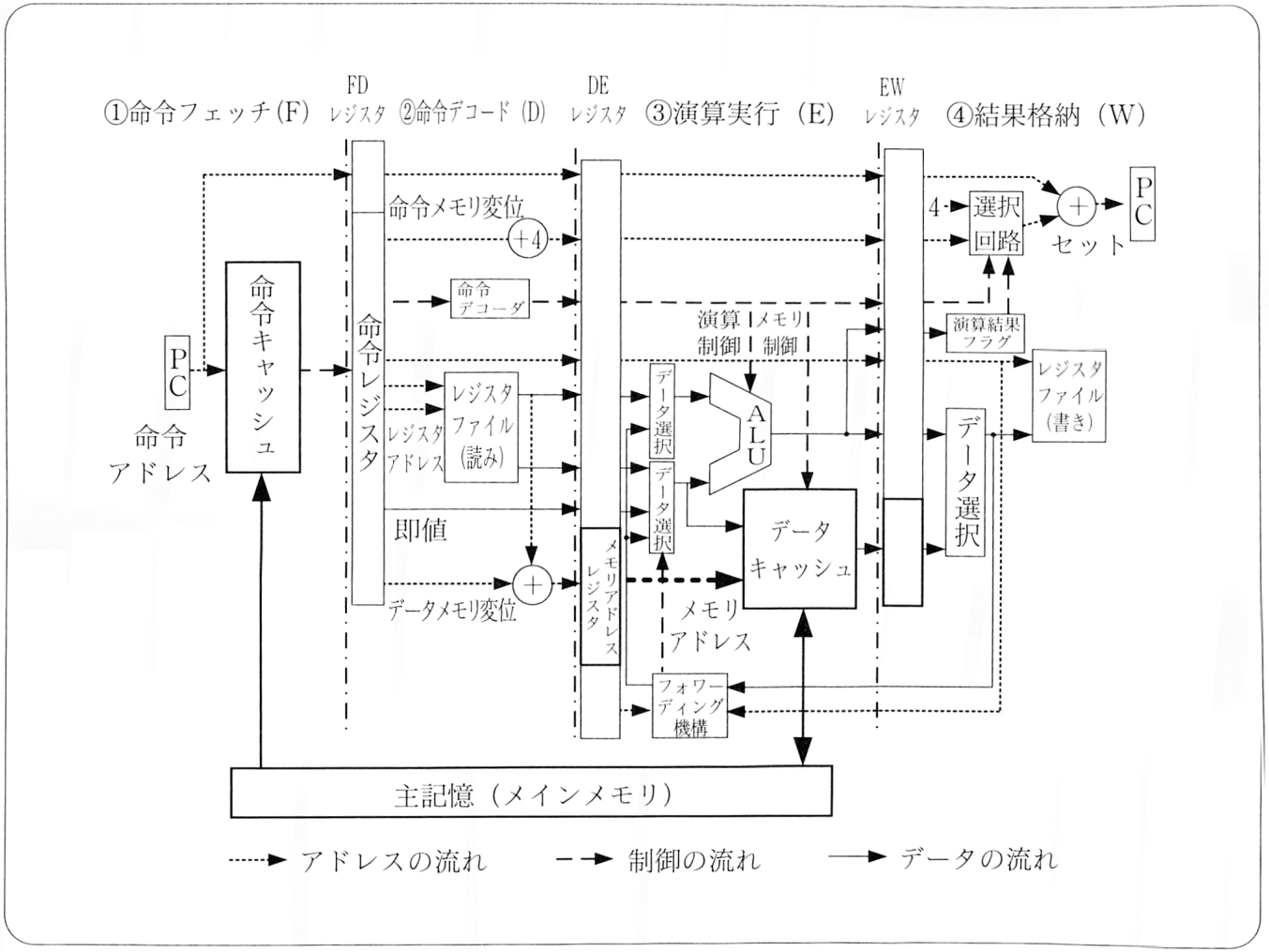


図 5.9　キャッシュの入ったパイプライン

5.2.7 キャッシュの性能

- プログラムの実行時間を，CPUが動いている時間と，メモリがストールしている時間に分ける.

$$T_p = T_{cpu} + T_{mstall}$$

- プログラムの命令数N，ロードストア命令の割合r_{ls}，メモリストールは全てキャッシュミスによって起こると考え，ミスの割合r_{miss}，1回のミス当たりのストール時間t_{mstall}，クロックをC [Hz]とすると.

$$T_p = N \frac{1 + r_{ls} \cdot r_{miss} \cdot t_{mstall}}{C}$$

が成り立つ. t_{mstall} は主記憶の速度で決まる.

例題5.1

$r_{ls} = 0.3$ のとき，下記のミス率，ミスペナルティで速度はどの程度落ちるか？

表5.3　キャッシュミス率とミスペナルティの例

事　例	ミス率	ミスペナルティ	実行時間相対値
事例1	0	——	1
事例2	0.05	10	——
事例3	0.05	50	——
事例4	0.5	10	——
事例5	0.5	50	——

$\frac{N}{C}$ 本来の速度　　$1 + r_{ls} \cdot r_{miss} \cdot t_{mstall}$ 実行時間相対値

例題5.1

表5.4　キャッシュミス率とミスペナルティの例

事　例	ミス率	ミスペナルティ	実行時間相対値
事例1	0	——	1
事例2	0.05	10	1.15
事例3	0.05	50	1.75
事例4	0.5	10	2.5
事例5	0.5	50	8.5

5.3 仮想記憶

5.3.1 仮想記憶とは何か

- 低速大容量の補助記憶装置（二次記憶）を利用して，主記憶の容量を大きく見せるための透過的な仕組み.

5.B 仮想記憶の原理			
仮想アドレス(virtual address)	⇒	物理アドレス(physical address)	[変換]
二次記憶のデータ	⇔	主記憶のデータ	[コピー，スワップ]

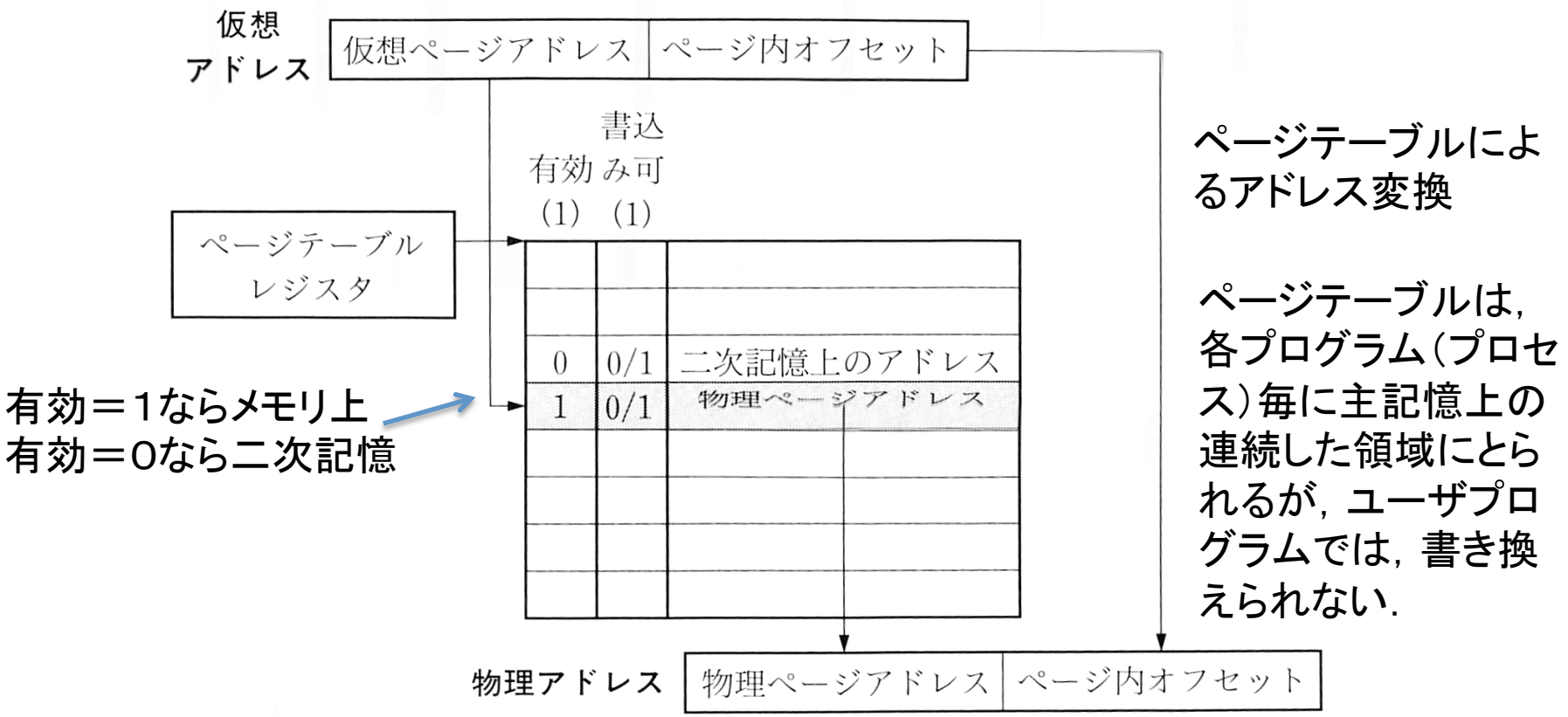
効果

① 大きなメモリを要するプログラムが書けるようになる.

② 複数のプログラムが1つの物理記憶を安全に分け合って使えるようになる.

5.3.2 仮想記憶の構成

- アドレス変換と，ページスワップ機構.
- ページスワップは時間がかかるので，ページのミスが発生しにくいフルアソシアティブ方式を採用.



5.3.3 ページフォールト

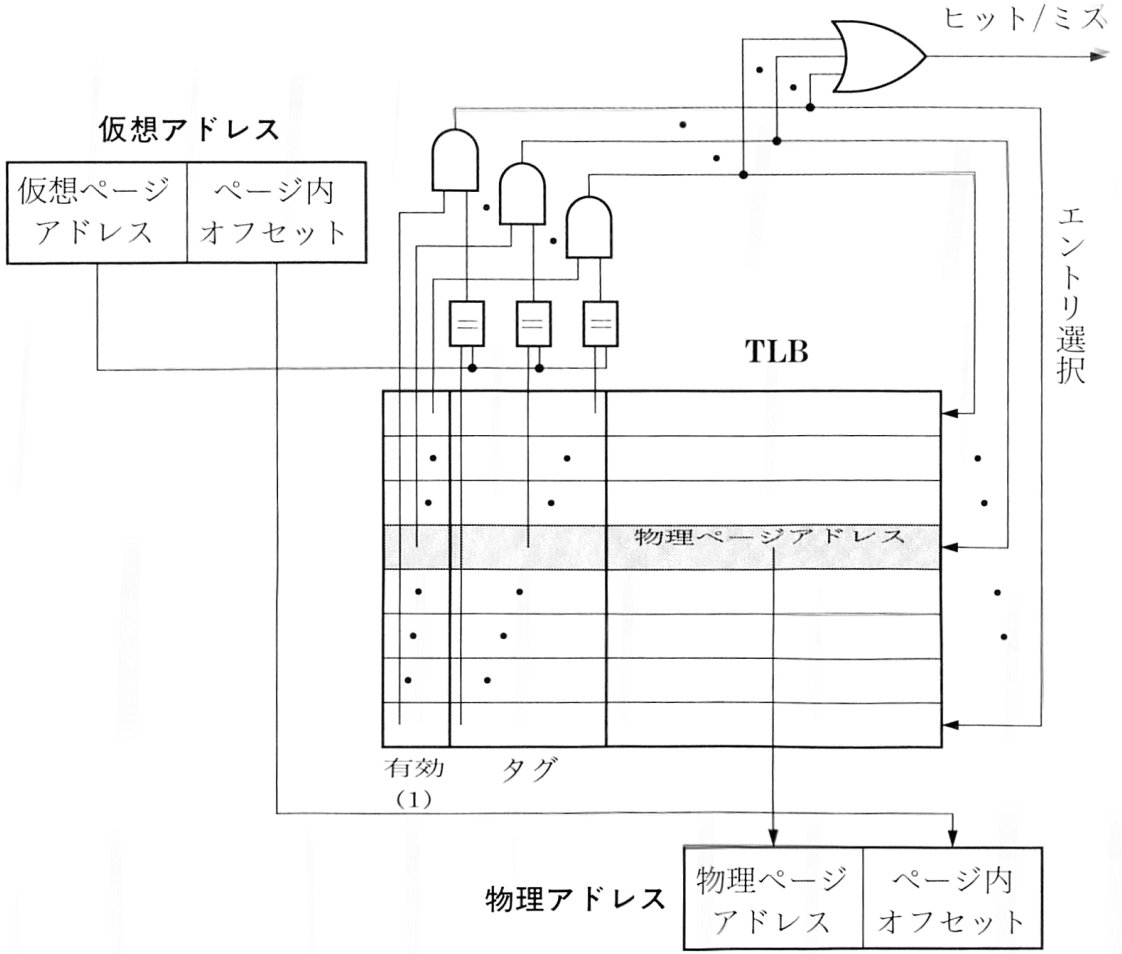
有効ビット=0のページは主記憶上ではなく，二次記憶の上に存在する．ここのページに対してアクセスが起きることを「ページフォールト」と呼ぶ．

【ページフォールト時の処理手順】

1. CPUの処理を中断
2. もし，主記憶に空き領域がなければ，主記憶上のページを2次記憶に書き出す．ページテーブルも更新．
3. 二次記憶上のページデータを主記憶に転送する．
4. ページテーブルの物理アドレスを書き換え，有効ビットを1にする．
5. CPUの処理を再開する．

5.3.4 TLB

- ページテーブル専用のキャッシュ（translation lookaside buffer）フルアソシアティブのキャッシュ



1. メモリアクセスが起きると仮想ページアドレスをタグとしてTLBの参照が起きる.
2. TLBがヒットすると，物理ページアドレスが取り出され，ページ内オフセットと組み合わせて物理アドレスが生成される.
3. TLBがミスすると，ページテーブルが参照され，その結果がTLBの空き部分に書き込まれる.
4. 空いていない場合は空きが作られる.

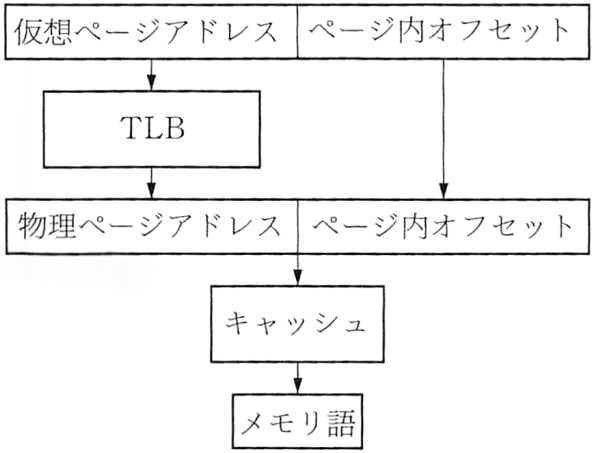
５．４　メモリアクセス機構

5.4.1 キャッシュと仮想記憶

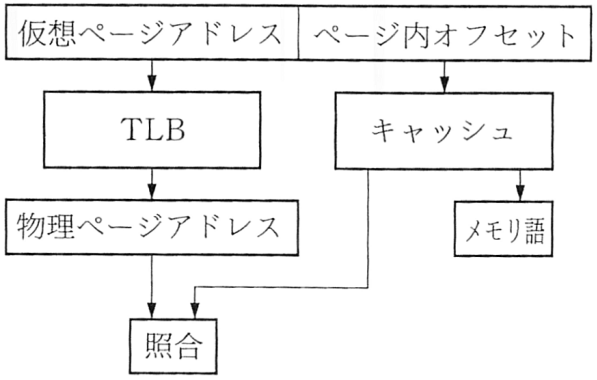
- キャッシュと仮想記憶の組み合わせ方

ページテーブルの参照は省略して記述してある.

速度的には遅いが，キャッシュサイズに制限がなく，エイリアスの問題も発生しない.



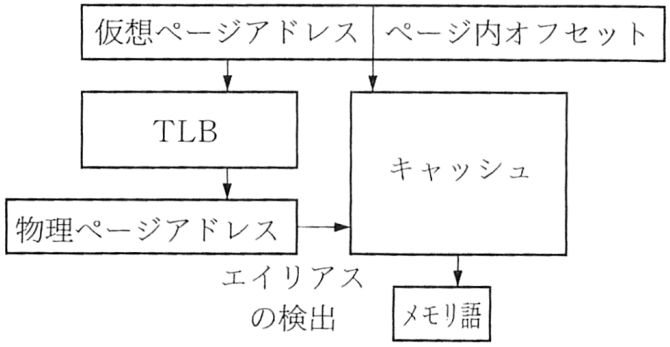
（a） 直列形物理アドレスキャッシュ



（b） 並列形物理アドレスキャッシュ

速度的には速く，エイリアスの問題も発生しないが，キャッシュのサイズがページ内オフセットに制限される.

速度的には速く，キャッシュのサイズにも制限はないが，二つの仮想アドレスが一つの物理アドレスを指してしまう現象（エイリアス）が発生する.



（c） 仮想アドレスキャッシュ

キャッシュサイズを特に大きくしたいなどの要求名なければ，へいれつ物理アドレスキャッシュが良いとされる.

5.4.2 メモリアクセス機構

- 並列物理アドレスキャッシュ
- 仮想ページアドレスを対象としたTLBはフルアソシアティブ
- ページ内オフセットについては，2ウェイセットアソシアティブキャッシュ

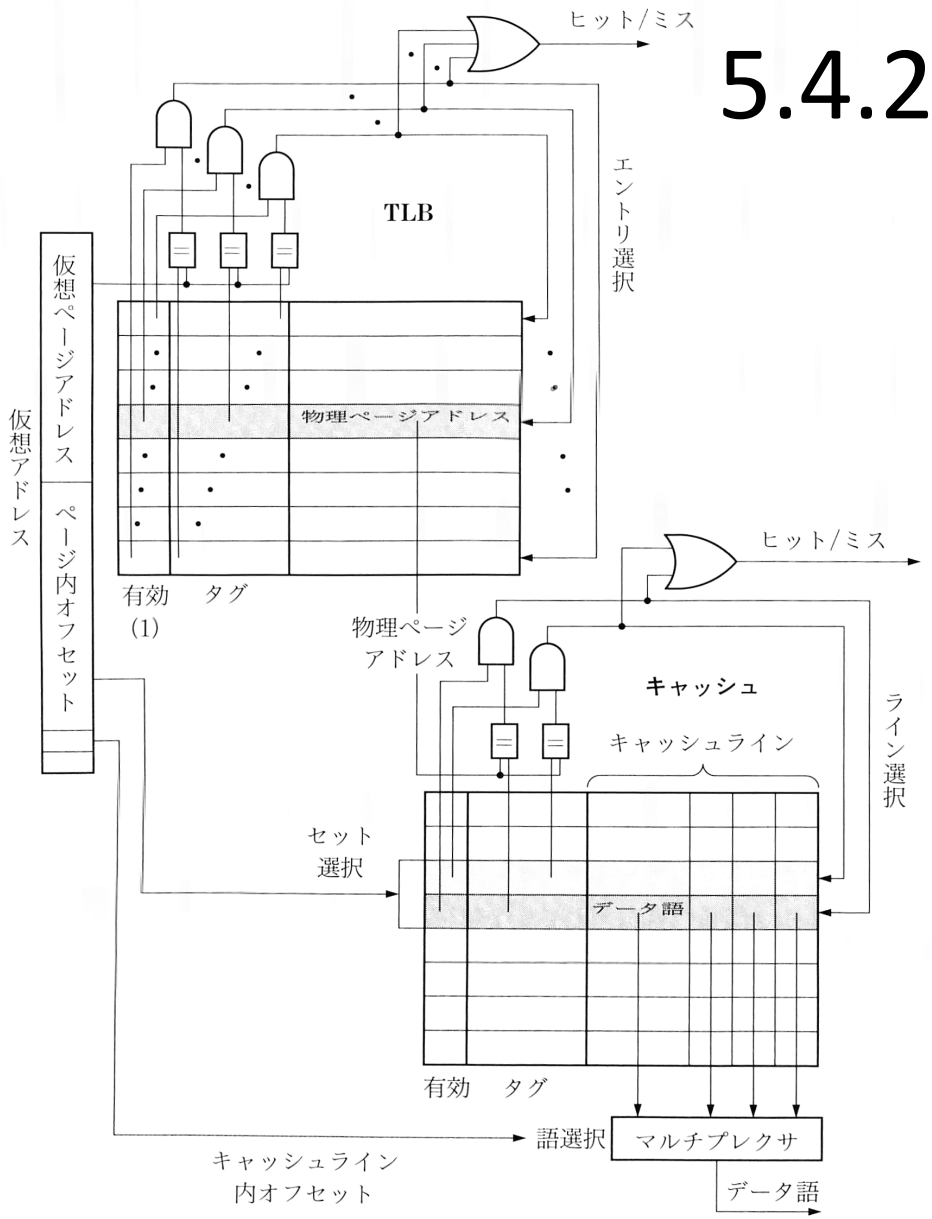


図 5.13　キャッシュと仮想記憶の入ったメモリアクセス機構

本日の講義の範囲