計算機方式論

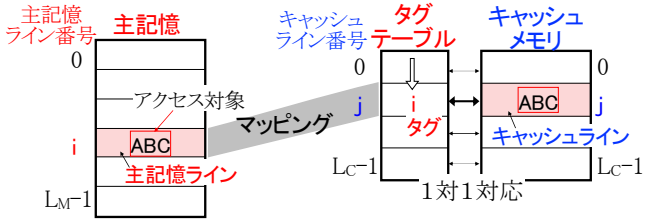
第13章 キャッシュ

−マッピング、書込アクセス−

1

キャッシュと主記憶とのマッピング

- 主記憶ラインｉとキャッシュラインｊとの対応付けは、タグテーブルなるマッピング表を使った連想写像方式（フルアソシアティブマッピング、セットアソシアティブマッピング）などで行う。
主記憶ラインｉがキャッシュラインｊにコピーされているとき、タグテーブルの第jエントリにはタグi（主記憶ライン番号）が入る。



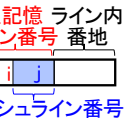
ラインサイズ:S_L　主記憶容量:S_M　キャッシュ容量:S_C
主記憶ライン数:$L_M=S_M/S_L$　キャッシュライン数:$L_C=S_C/S_L$

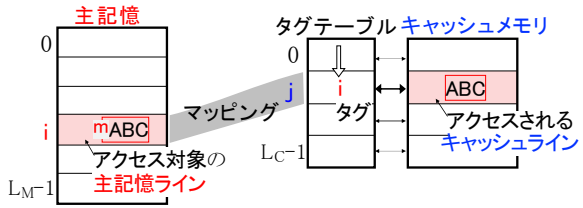
2

キャッシュラインの決定（連想写像方式の場合）

(1)アクセスする主記憶番地mから、主記憶のライン番号iを求め（$i=m/S_L$,ライン内番地$=m\%S_L$）、iをタグとして、タグテーブルを連想検索（フルアソシアティブマッピング）する。

(2)①タグiと一致するエントリjがあれば、ヒットで、キャッシュラインjをアクセス。
②タグiと一致するエントリがなければ、ミスヒットで、ミスペナルティ処理を行う。





3

マッピング方式

①ダイレクトマッピング(direct mapping)

②フルアソシアティブマッピング
(full associative mapping)

③セットアソシアティブマッピング
(set associative mapping)

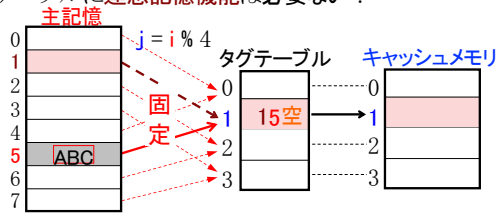
4

①ダイレクトマッピング

- 主記憶ライン i に対するキャッシュライン j は予め一意に定めておく。
  例えば、 $j = i \% L_c$ としたとき、
  ① タグテーブル j番目に主記憶ラインのタグiが在るときは、ヒットで、
  キャッシュライン j をアクセス。
  ② j番目が空きのときは、ミスヒットで、主記憶ライン i をロードし、
  タグiを書き込む。
  ③ j番目にiと違うタグが在るときも、ミスヒットで、そのキャッシュラインを追い出した後、主記憶ライン i をロードし、タグiを書き込む。

×ライン置換の選定の自由度が低い。

◎タグテーブルに連想記憶機能は必要ない！



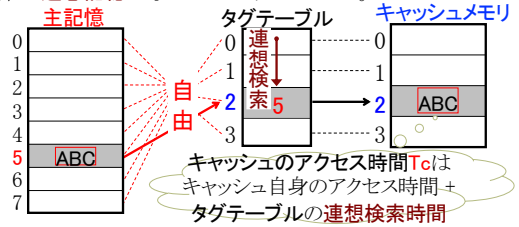
②フルアソシアティブマッピング

- 主記憶ラインをどのキャッシュラインにも自由にマッピング。
  タグテーブル中、主記憶ラインのタグを連想検索で探す。
  タグが見つかれば、ヒットでアクセス。なければミスヒットで、
  空きキャッシュラインがあれば、主記憶ラインをロードし、タグテーブルにタグを書き込む。
  空きがなければ、ライン置換アルゴリズムで追い出すキャッシュラインを決め、そこに主記憶ラインをロードし、タグテーブルにタグを書き込む。

◎ ライン置換アルゴリズムの選定の自由度が高い。

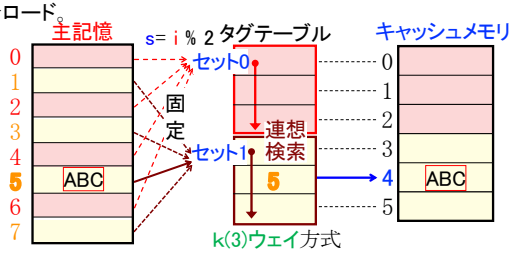
× タグテーブル全体に連想記憶機能が必要なため、実装コストが高い。

× タグ検索に連想記憶のオーバーヘッドが生じる。



③セットアソシアティブマッピング

- ダイレクトマッピングとフルアソシアティブマッピングを組み合わせた方式。
- キャッシュメモリをN個のセットに分割し、主記憶ライン i からセットへのマッピングはダイレクト方式で行う。例えば、セット番号 $s = i \% N$。
- 各セットは、対応するタグテーブルに連想記憶機能をもたせてあり、セット s 中をタグiで連想検索で探す。
- タグiが見つかればヒットでアクセス、見つからなければミスヒットで、空きキャッシュラインがあれば主記憶ラインをロード、空きがなければ、ライン置換アルゴリズムで追い出すキャッシュラインを決め、そこに主記憶ラインをロード。

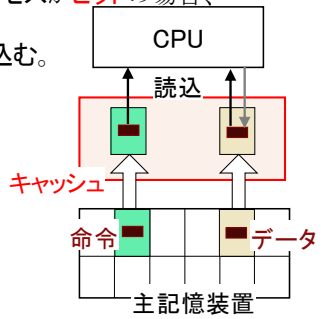


kウェイセットアソシアティブマッピング

- 1セットをk個のキャッシュラインで構成するとき、kウェイセットアソシアティブマッピング(k-way set associative mapping)という。
  kは2～16程度（前図は3ウェイ）
- キャッシュメモリのセット数 N = Lc/k
- N個の連想記憶タグテーブルが必要だが、フルアソシアティブマッピングと比べ、
- テーブルのエントリの数はN分の1のため、
  検索時間が短く、ハードウェア実装コストが格段に下がる。
- k=1ウェイ,N= Lcセットの場合、ダイレクトマッピング。
- k=Lcウェイ,N=1セットの場合、フルアソシアティブマッピング。

キャッシュの読込アクセス

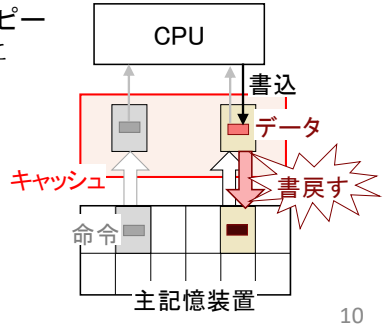
- 命令キャッシュは、読込アクセスだけ。
- データキャッシュは、読込と書込の2つのアクセス。
- 命令やデータの読込アクセスがヒットの場合、キャッシュラインからその命令・データを読み込む。



9

キャッシュの書込アクセス

- データの書込アクセスがヒットの場合、キャッシュの内容を書き換える（対象のデータの書き換え）
- キャッシュは主記憶のコピーなので、変更を主記憶に反映するため、主記憶に書き戻す！これを、主記憶の更新という！



10

主記憶の更新

- 「主記憶ラインとキャッシュラインとの内容の同一性」をコヒーレンシ(coherency)といい、これを保つため、キャッシュ制御機構は、書込アクセスのヒット時には主記憶更新（主記憶ラインへの書き戻し）を行わなければならない。
- いつの時点で主記憶ラインを更新するかで、2つの方式がある。

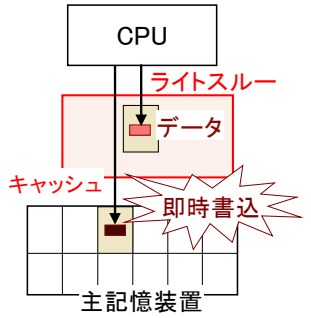
  ①ライトスルー（ストアスルー）
  キャッシュの書き込みと同時に主記憶更新。

  ②ライトバック（ストアバック,コピーバック）
  キャッシュの書き込み時には主記憶更新しないで、当該キャッシュラインが追い出しの対象になったとき、主記憶更新。

11

①ライトスルー(write-through)

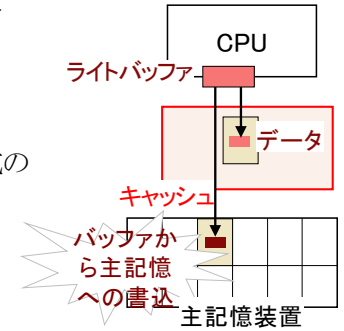
- キャッシュの書き込みと同時に主記憶にも書き込む（主記憶更新する）。
- コヒーレンシを常に保証し、主記憶更新のタイミングを図らなくてよい。
- 書込アクセスが、実質的に主記憶アクセスなので、キャッシュ効果があがらず、書込アクセスの性能が改善されない。



12

ライトスルー（置いてきぼり制御）

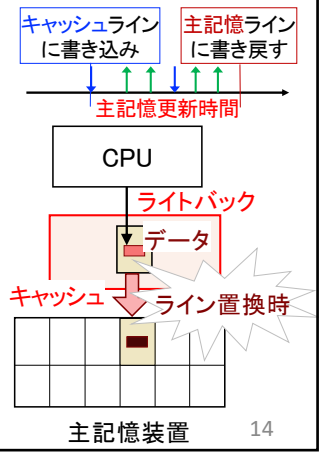
- プロセッサに**ライトバッファ**(キュー)を置き、書込データをバッファに一時記憶する。
- 主記憶へのアクセスを**後回し**にするため**高速化**が望める。
- バッファの読み出し、マルチプロセッサ方式のときに問題が生じる。



13

②ライトバック(write-back)

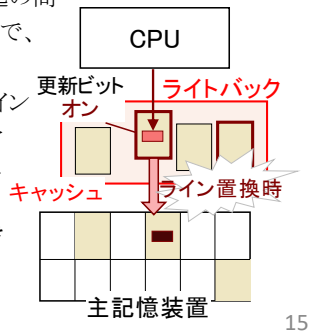
- **キャッシュの書き込み**時には、**主記憶更新しない**。
- 書き込み対象を含むラインを**主記憶を更新**する**時点**は、
- **アソシアティブマッピング方式**では、**空きライン**が必要になり、**ライン置換アルゴリズム**に**当該キャッシュライン**が選ばれ、**追出し対象**になったとき（**ライン置換時**）。
- **ダイレクトマッピング方式**では、**当該キャッシュライン**が選ばれたが、**キャッシュタグが異なる**とき。



14

ライトバック−更新ビット

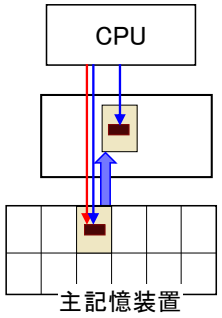
- **ライトバック**は、**ヒット時**には**キャッシュ性能を十分発揮**でき、**主記憶へのアクセス回数を減らす**ことができる。
- **キャッシュ更新**から**主記憶更新**迄の間は**コヒーレンシが壊されている**ので、それを管理し保つ機構が必要。
- **タグテーブル**にそのキャッシュラインへの書き込みを示す**更新ビット**を用意し、**書き込まれたら“オン”**にする。**ライン置換時**に**追い出し対象に選ばれたら**、**“オン”のとき**だけ主記憶ラインに**書き戻し**、**“オフ”**のときは**書き戻さない**。



15

書込アクセスがミスヒットの場合

- **ノーライトアロケート**(no write allocate)
  主記憶への書き込みだけで、主記憶ラインの**キャッシュへの読み出しは行わない**。
- **ライトアロケート**(write allocate)
  当該の主記憶ラインを**キャッシュに読み出し**、つぎに**キャッシュ**と**主記憶**の**対象を書き換える**。



16