コンピュータアーキテクチャ（5）

坂井　修一
東京大学大学院 情報理工学系研究科 電子情報学専攻
東京大学　工学部　電子情報工学科／電気電子工学科

- はじめに
- キャッシュ

はじめに

- 本講義の目的
  - コンピュータアーキテクチャの基本を学ぶ
- 時間・場所
  - 火曜日 8:40 – 10:10　工学部2号館241
- ホームページ（ダウンロード可能）
  - url:　http://www.mtl.t.u-tokyo.ac.jp/~sakai/hard/
- 教科書
  - **坂井修一『コンピュータアーキテクチャ』（コロナ社、電子情報レクチャーシリーズC-9）**
  - **坂井修一『実践　コンピュータアーキテクチャ』（コロナ社）**

  教科書通りやります
- 参考書
  - D. Patterson and J. Hennessy, Computer Organization & Design、3rd Ed.（邦訳『コンピュータの構成と設計』(第3版)上下（日系ＢＰ））
  - 馬場敬信『コンピュータアーキテクチャ』（改訂２版）、オーム社
  - 富田眞治『コンピュータアーキテクチャＩ』、丸善
- 予備知識：　論理回路
  - 坂井修一『論理回路入門』、培風館
- 成績
  - 試験＋レポート＋出席

講義の概要と予定（１／２）

１．コンピュータアーキテクチャ入門

ディジタルな表現、負の数、実数、加算器、ＡＬＵ，フリップフロップ、レジスタ、計算のサイクル

２．　データの流れと制御の流れ

主記憶装置、メモリの構成と分類、レジスタファイル、命令、命令実行の仕組み、実行サイクル、算術論理演算命令、シーケンサ、条件分岐命令

３．命令セットアーキテクチャ

操作とオペランド、命令の表現形式、アセンブリ言語、命令セット、
算術論理演算命令、データ移動命令、分岐命令、アドレシング、
サブルーチン、RISCとCISC

４．パイプライン処理（１）

パイプラインの原理、命令パイプライン、オーバヘッド、構造ハザード、データハザード、制御ハザード

５．パイプライン処理（２）

フォワーディング、遅延分岐、分岐予測、命令スケジューリング

６．キャッシュ

記憶階層と局所性、透過性、キャッシュ、ライトスルーとライトバック、
ダイレクトマップ型、フルアソシアティブ型、セットアソシアティブ型、キャッシュミス

7．仮想記憶

仮想記憶、ページフォールト、TLB、物理アドレスキャッシュ、仮想アドレスキャッシュ、メモリアクセス機構

講義の概要と予定（２／２）

８．基本CPUの設計

ディジタル回路の入力、Verilog HDL、シミュレーションによる動作検証、アセンブラ、基本プロセッサの設計、基本プロセッサのシミュレーションによる検証

９．命令レベル並列処理（１）

並列処理、並列処理パイプライン、VLIW、スーパスカラ、並列処理とハザード

１０．命令レベル並列処理（２）

静的最適化、ループアンローリング、ソフトウェアパイプライニング、トレーススケジューリング

１１．アウトオブオーダ処理

インオーダーとアウトオブオーダー、フロー依存、逆依存、出力依存、
命令ウィンドウ、リザベーションステーション、レジスタリネーミング、
マッピングテーブル、リオーダバッファ、プロセッサの性能

１２．入出力と周辺装置

周辺装置、ディスプレイ、二次記憶装置、ハードウェアインタフェース、割り込みとポーリング、アービタ、DMA、例外処理

試験：　7月

前回のまとめ

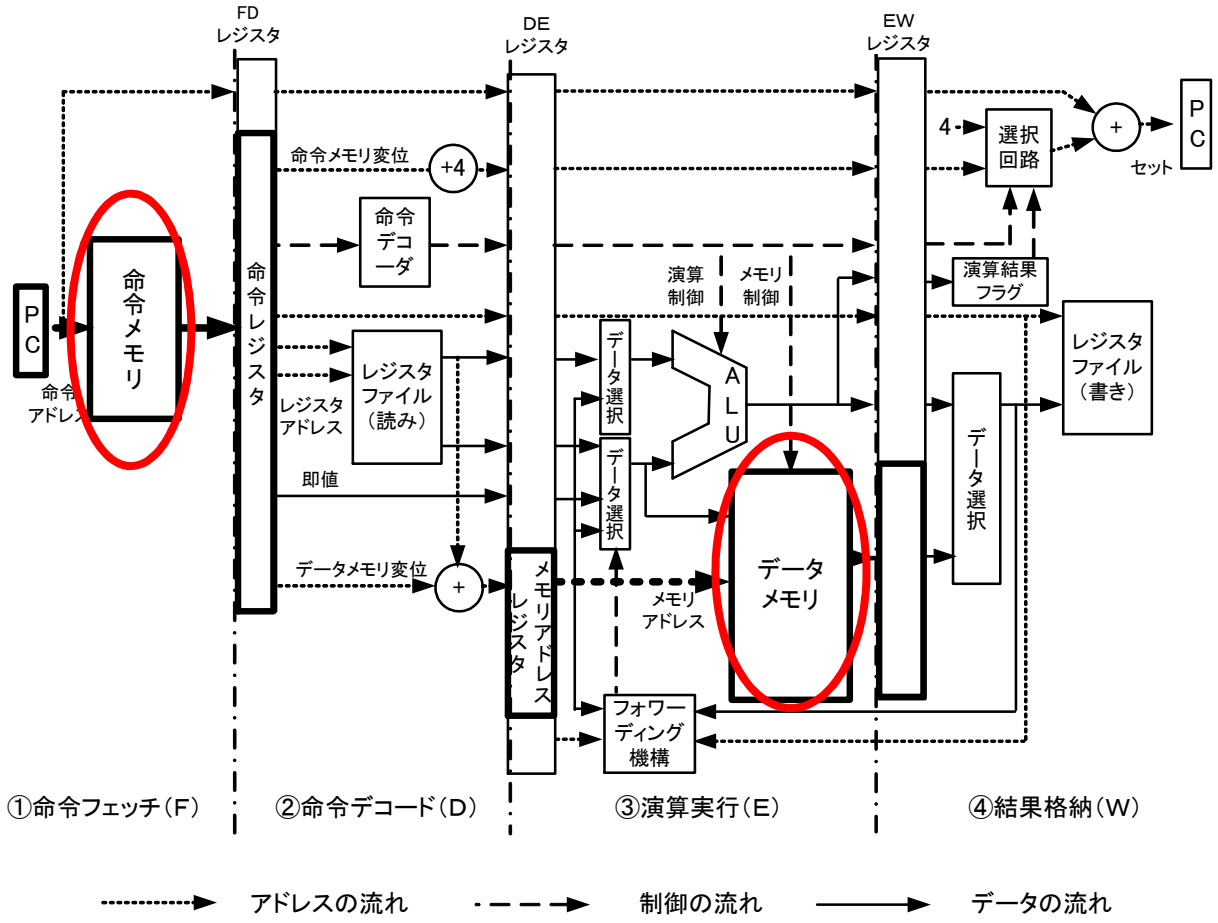
- パイプライン
  - 流れ作業によって処理効率を飛躍的に向上させる技術
- 処理時間
  - 一つの命令の処理が始まってから完了するまでの時間。
  - NxT（Nはパイプラインのステージ数、Tは1ステージでの処理時間）
- スループット
  - 単位時間に終了する処理量（命令数）1／T
- 基本命令パイプライン
  - 命令フェッチ（F),命令でコード（D),演算実行（E)、結果の格納（W)からなるコンピュータのパイプライン（他のアーキテクチャではオペランド読み出し（R)やメモリアクセス（M)があるものもある。）
- パイプライン阻害要因
  - 最も時間のかかるステージ、パイプラインレジスタによる遅延、ハザード
- ハザード
  - パイプライン動作ができなくなる状態。構造ハザード、データハザード、制御ハザードに分類される。
- 構造ハザード
  - コンピュータの内部構造に原因をもつハザード
- データハザード
  - 命令間のデータ依存関係に基づくハザード
- 制御ハザード
  - 分岐命令によるハザード
- フォワーディング
  - 前（前々）の命令の結果を直接Eステージに送る事でデータハザードを解消する手法
- 制御ハザードの緩和法
  - 命令アドレスの早期生成、遅延分岐、分岐予測、命令スケジューリング
- 分岐予測
  - 分岐の有無を予測し、成功すれば続行、失敗すればパイプラインをフラッシュする。実装方法としては、2ビット予測器、2レベル適応型予測など

7. キャッシュ

- 内容
  - 記憶階層
    - 命令パイプラインとメモリ
    - 記憶階層と局所性
    - 透過性
  - キャッシュ
    - キャッシュとはなにか
    - ライトスルーとライトバック
    - ダイレクトマップ型キャッシュ
    - キャッシュミス
    - フルアソシアティブ型キャッシュ
    - セットアソシアティブ型キャッシュ
    - キャッシュの入ったCPU
    - キャッシュの性能

命令パイプラインとメモリ

- 問題： はたしてメモリは1クロックで読み書きできるか？



命令メモリ、データメモリは
ＣＰＵチップの外に置かれる
↓
アクセスに時間がかかる

メモリの理想と現実

- 理想
  - 無限大の容量
  - 無限小のアクセス時間
  - 単純なアドレシング
- 現実
  - 大容量と高速アクセスは両立しない
    - 容量が大きくなるとアクセス時間も大きくなる

（メモ）メモリの階層化

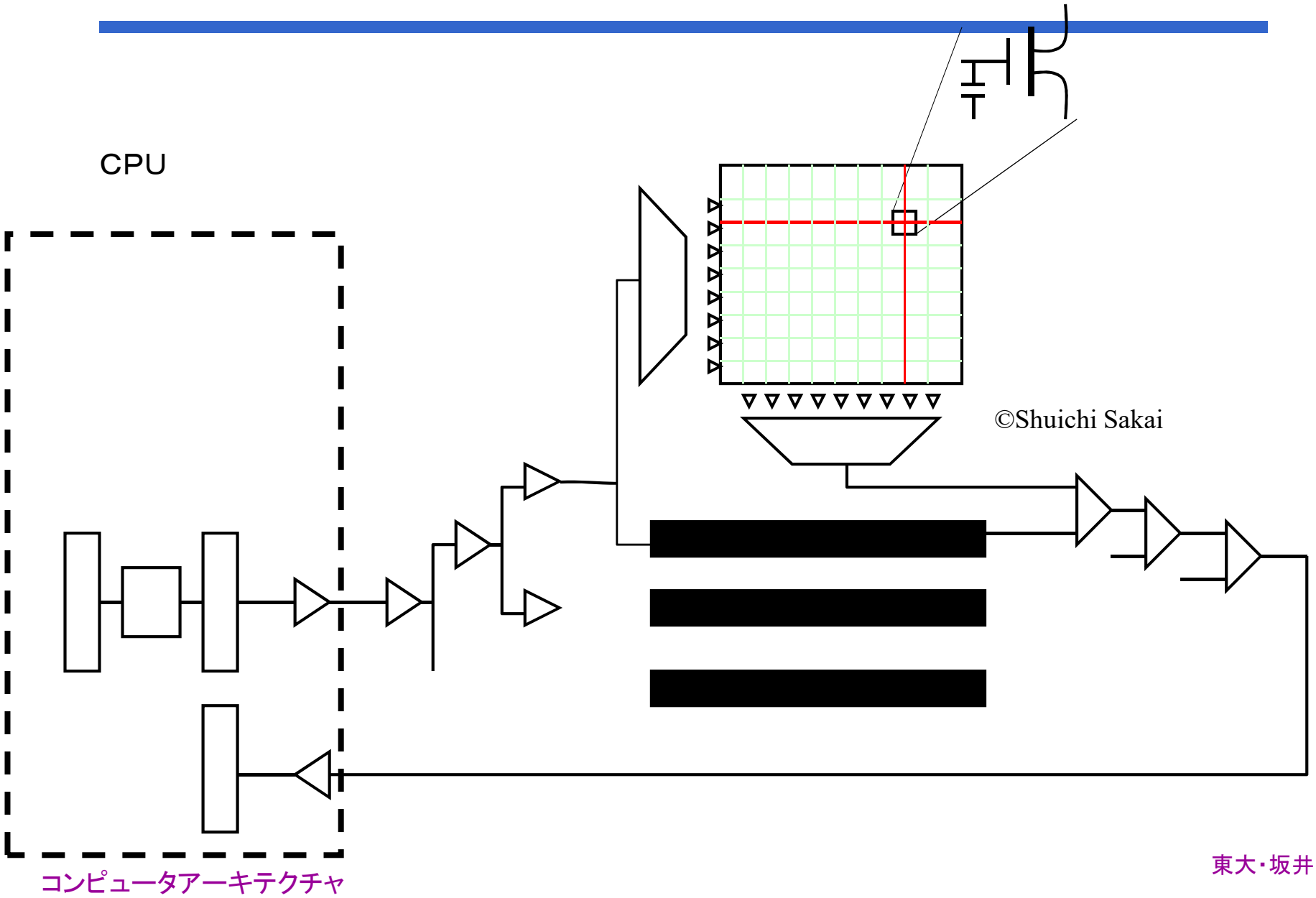
- アクセス・ギャップ
  - CPUの命令実行速度とメモリのアクセス時間のギャップの増大
    - メモリ素子技術とCPU素子技術の差異 （DRAMセルと、フリップフロップ）
    - チップ内遅延時間 （CPUは局所性利用、メモリは √n）
    - チップ内外のアクセス時間差 （チップ内 1GHz、チップ外 133MHz）
    - メモリの大規模化とアドレスTree，マルチプレクサ Tree
    - 例： CPU 1GHzクロック、1クロックに最大4命令実行

      メモリ SDRAM 133MHzクロック、アクセス時間6クロック

      メモリ DDR 266MHzクロック、アクセス時間10クロック

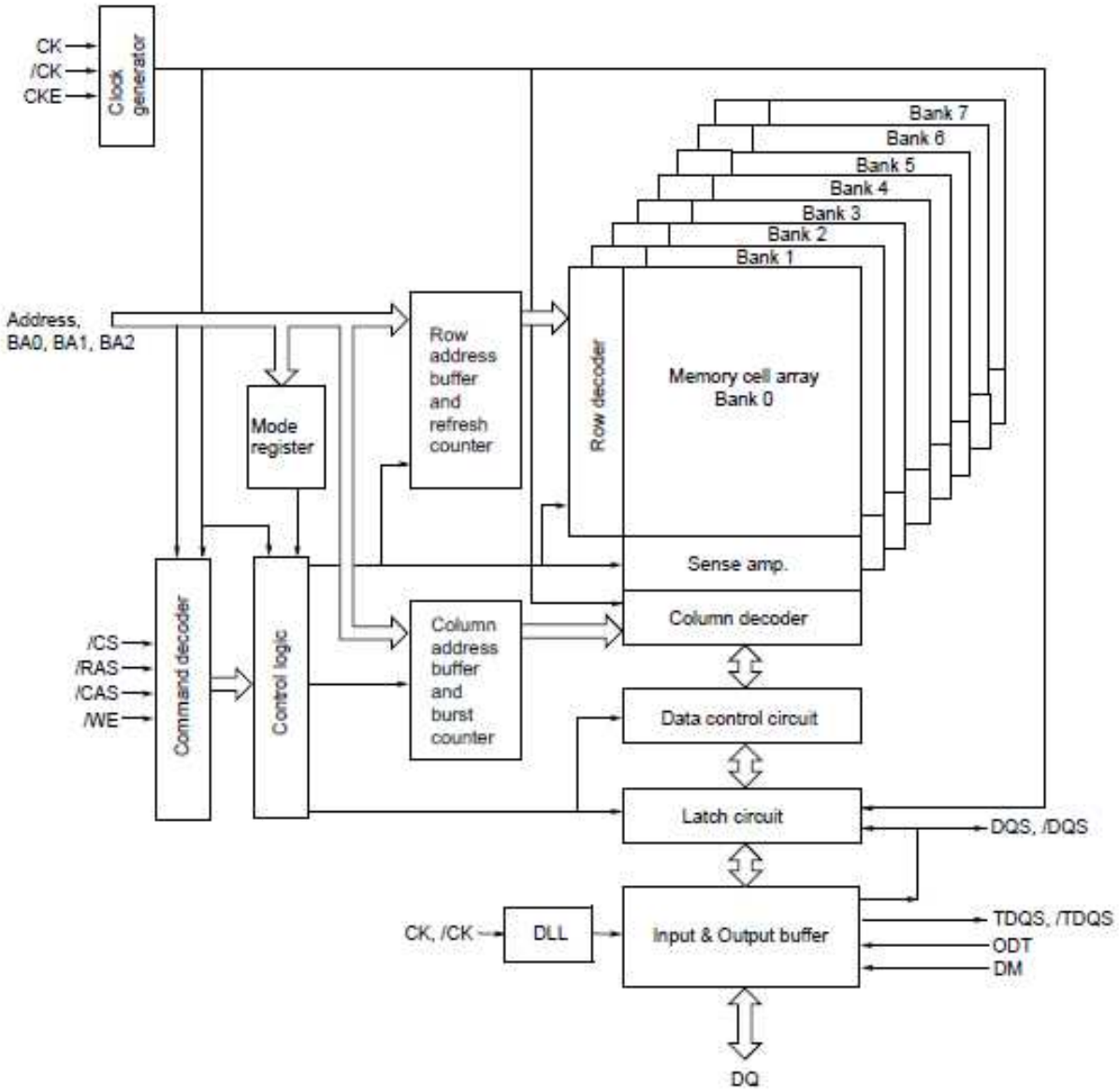
（DDR=Double Data Rate クロックの上げエッジと下げエッジの両方を用いる）

アクセスギャップ



DDR3メモリの内部構造

Block Diagram



DDR　SDRAMの性能

DDR2, DDR3メモリの性能パラメータ

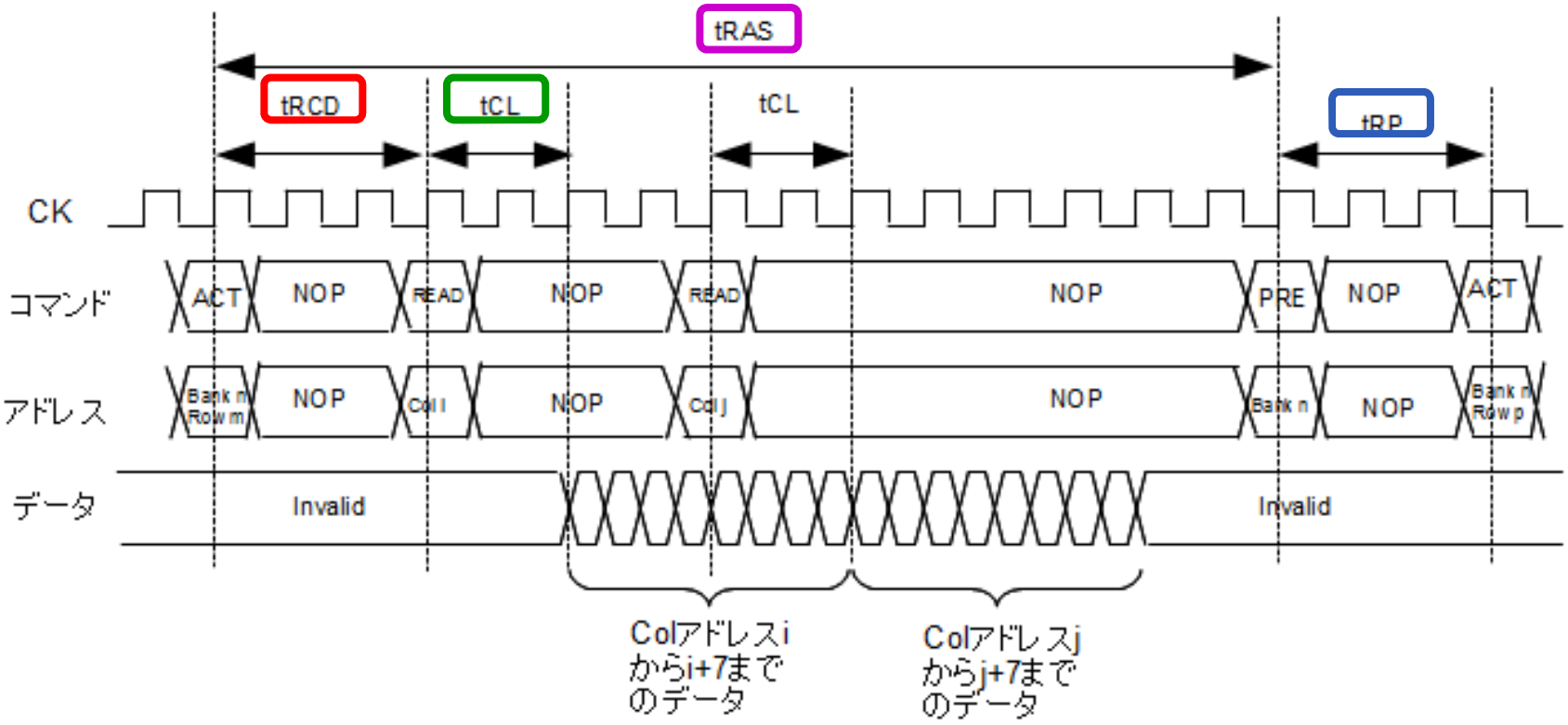
DDRx- nnnn (PC-xxxx)

レイテンシ　７－７－７－２４

をどのように解読するか？

CL-tRCD-tRP-tRAS

SDRAM（DDR)の読み出しタイミング



DRAMタイミングパラメータ　tRCD + tRP + tCL + tRAS

例：　DDR3-1600 PC12800　CLK=800Mz,　9-9-9-24

　　アクセス時間　22.5ns,　サイクル時間　41ns　CL-tRCD-tRP-tRAS

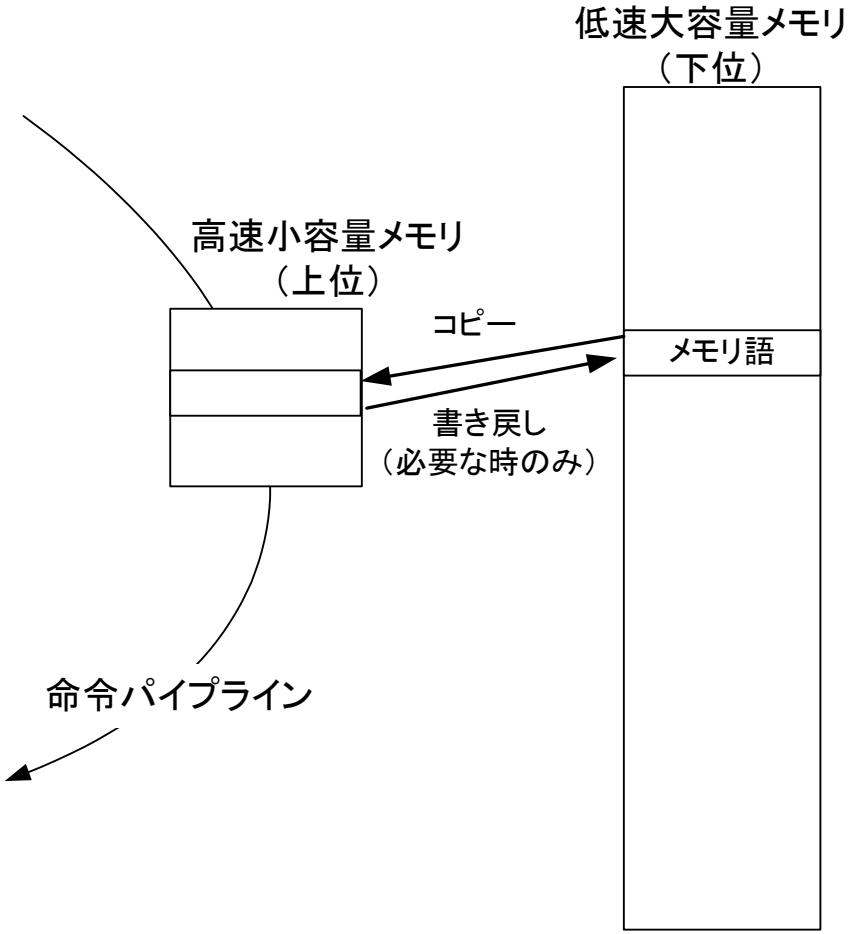
tRCD　　ACTコマンドからREADコマンドの最小サイクル数間隔

tRP　　PREコマンドから次のACTコマンドまでの最小サイクル数間隔

tCL　　READコマンドからデータが出力されるまでのサイクル数

tRAS　　ACTコマンドからPREコマンドの最小サイクル数間隔

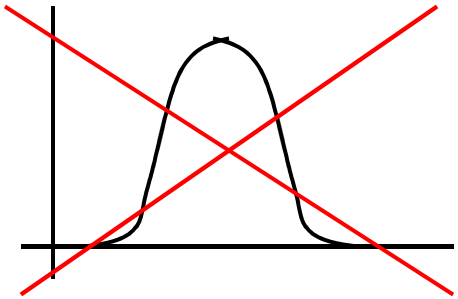
記憶階層と局所性



- なぜ記憶階層が有効か？
  命令やデータに局所性があるから
  - 空間局所性
    - あるメモリ語が参照されたときに、その語の近くの語が引き続き参照される性質
  - 時間局所性
    - あるメモリ語が参照されたとき、その語が時間をおかずに再び参照される性質

メモ：メモリアクセスの局所性

- 決して、あるアドレスを中心とした確率分布でない



- メモリアクセスの基本パターン
  - 命令アクセス
    - ループ構造、再帰構造
    - 入れ子構造（サブルーチン）
    - 例外的処理
  - データアクセス
    - ベクトル，マトリクス
    - リスト構造
    - スタック、ローカル変数

メモ：時間的局所性と空間的局所性

- 時間的局所性(Temporal Locality)
  - 一度アクセスしたアドレスに、近い将来再びアクセスすること
  - ループ内のスカラ変数、大域変数など
  - ループ構造、再帰構造の命令アクセス

- 空間的局所性(Spacial Locality)
  - ある場所にアクセスすると、近い将来その近傍の場所にアクセスすること
  - ベクトル、マトリクスのアクセス
  - 逐次実行中の命令アクセス
  - 入れ子構造のローカルデータ

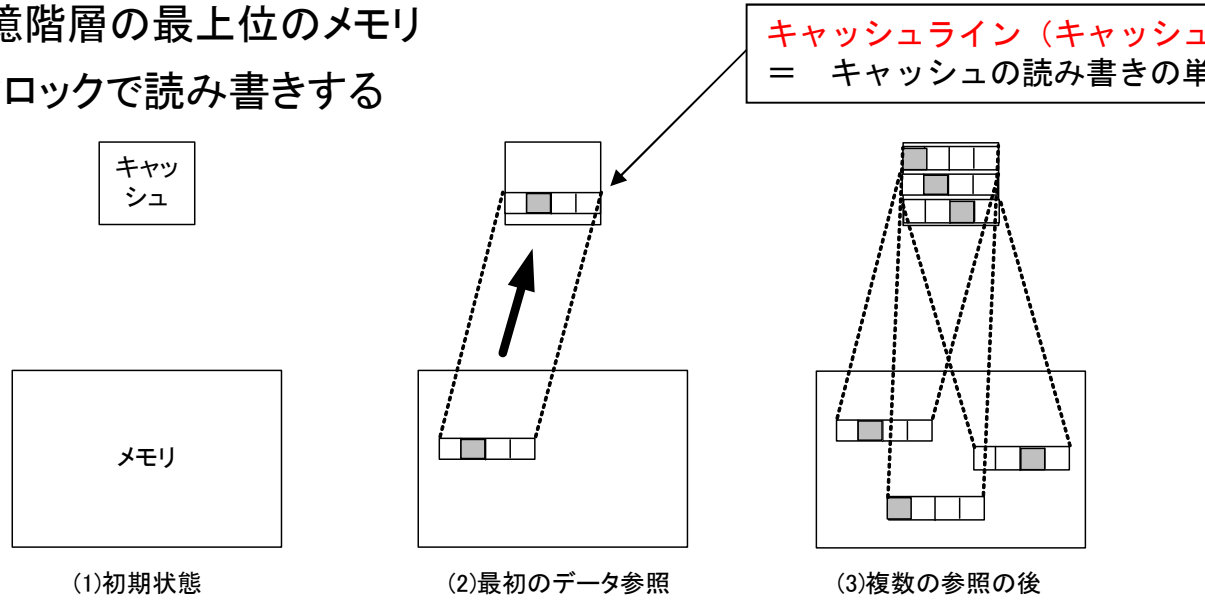
- 一般のプログラムではこの両者が混合して出現
- 全アドレス空間中に、局所性を持つ場所がN個出現
  - 一般に、Nは2から5位（強くプログラム依存）

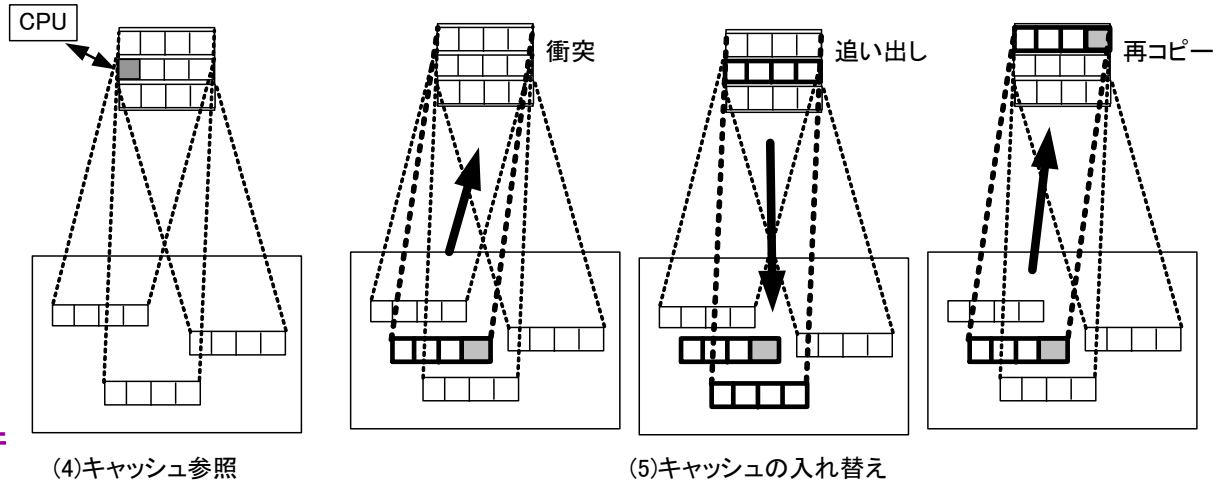
記憶階層と機械語プログラム

- 記憶階層を陽に見せる方式
  - 各階層のメモリのそれぞれにアドレスを付け、コピーや書き戻しもロード命令・ストア命令で実現する
  - ×アセンブリ言語のプログラマが記憶階層を意識しなくてはならず、いつも最良のメモリの利用法を考えてプログラムをしなくてはならない。
  - ×命令セットが同じでもメモリの階層構成が変化したり、各階層のメモリの大きさが変化したりするたびにプログラムを作り直さなくてはならない。
- 記憶階層を見せない方式
  - 見かけ上は、高速で大容量のメモリが1つだけあるものとして機械語のプログラムを書き、ハードウェアの機構でどの階層のメモリをどう使って局所性を活かすかを決める
  - ◎単純に命令セットだけを意識して機械語プログラムを書いておけば、効率や安全性はハードウェアが勝手に面倒を見てくれる。この性質を**透過性(transparency)**と呼ぶ。

キャッシュとは何か

- キャッシュ
  - 記憶階層の最上位のメモリ
  - 1クロックで読み書きする

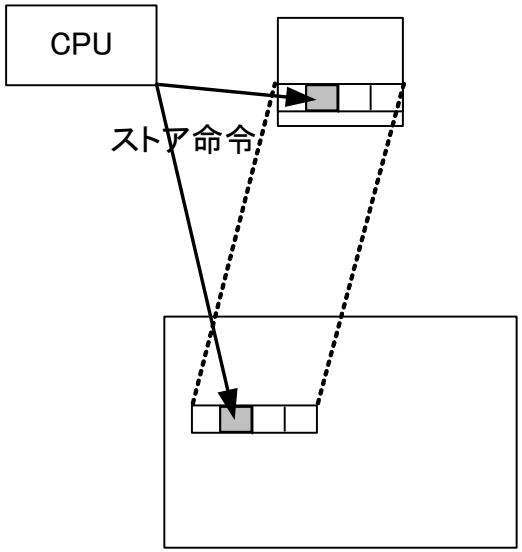




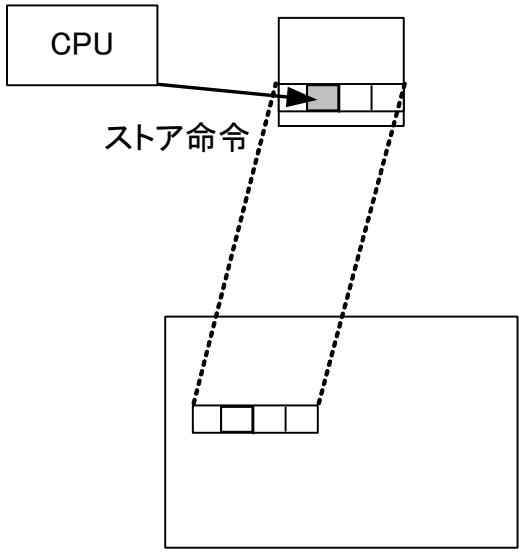
キャッシュの分類

- メモリ書き込み方式による分類
  - ライトスルー
  - ライトバック
- 連想度による分類
  - ダイレクトマップ
  - フルアソシアティブ
  - セットアソシアティブ

ライトスルー方式とライトバック方式



(1)ライトスルー方式



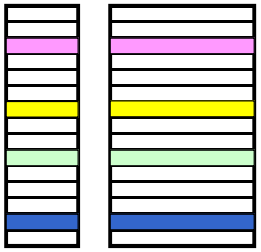
(2)ライトバック方式

表 5.1 ライトスルー方式とライトバック方式

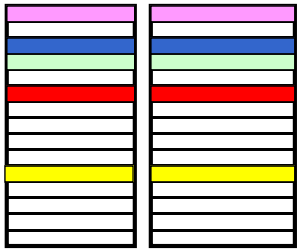
	ライトスルー方式	ライトバック方式
メモリアクセス	ストア命令の実行時	キャッシュライン追い出しの時
ライト命令の実行速度	ライトバッファの速度	キャッシュの速度
キャッシュライン書き戻し	不要	キャッシュライン追い出しの時
実装	単純	複雑

連想性

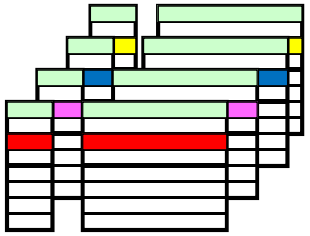
1. キャッシュブロックの探索



ダイレクト・マップ方式
インデクスで一意に

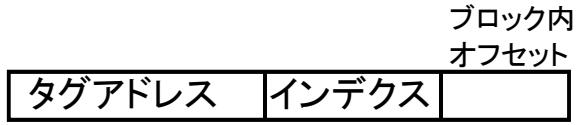


フルアソシアティブ方式

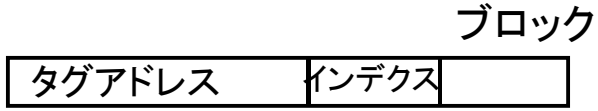


ブロック内オフセット　4　Way　セットアソシアティブ方式

ブロックアドレス

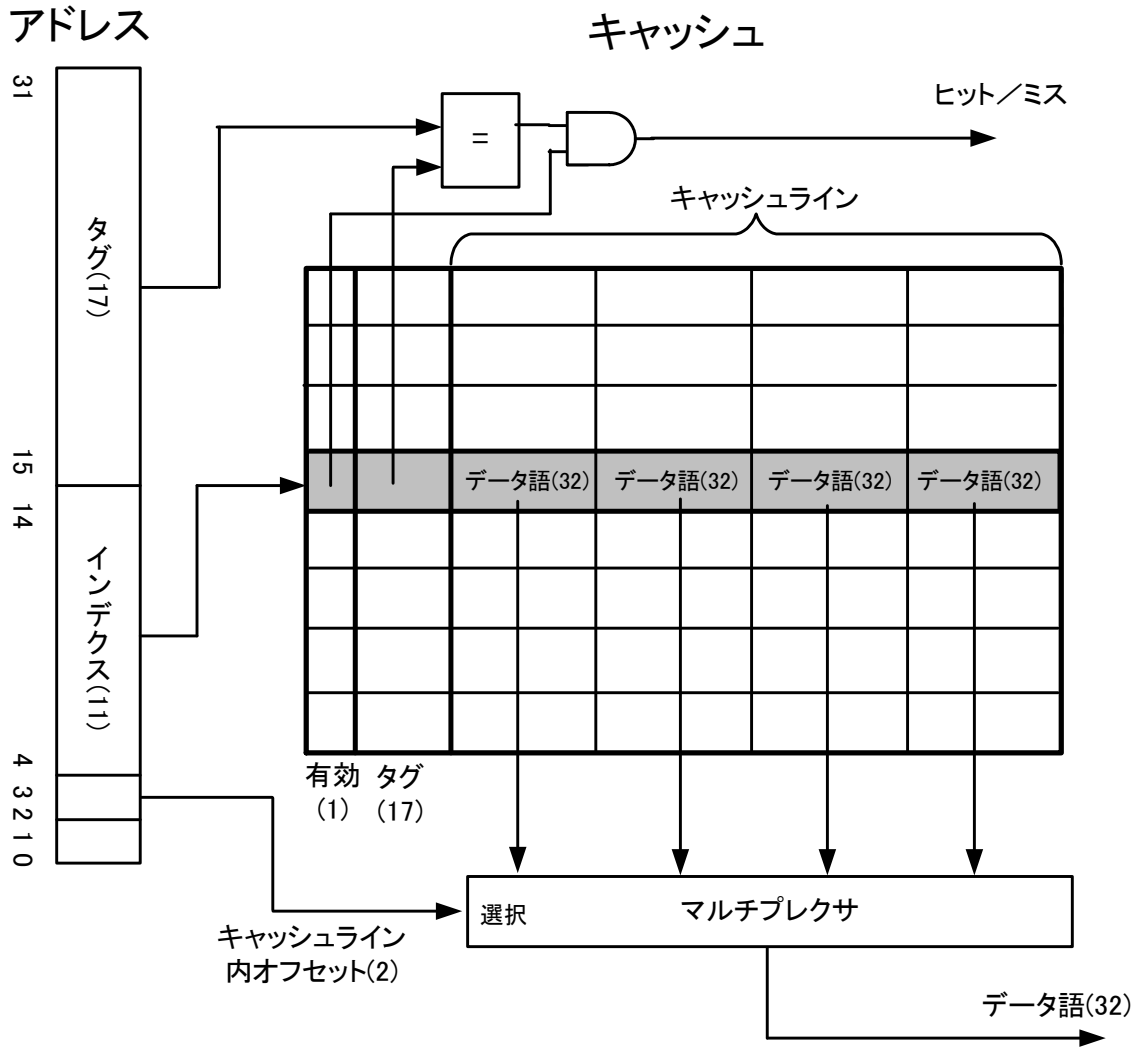


ブロックアドレス



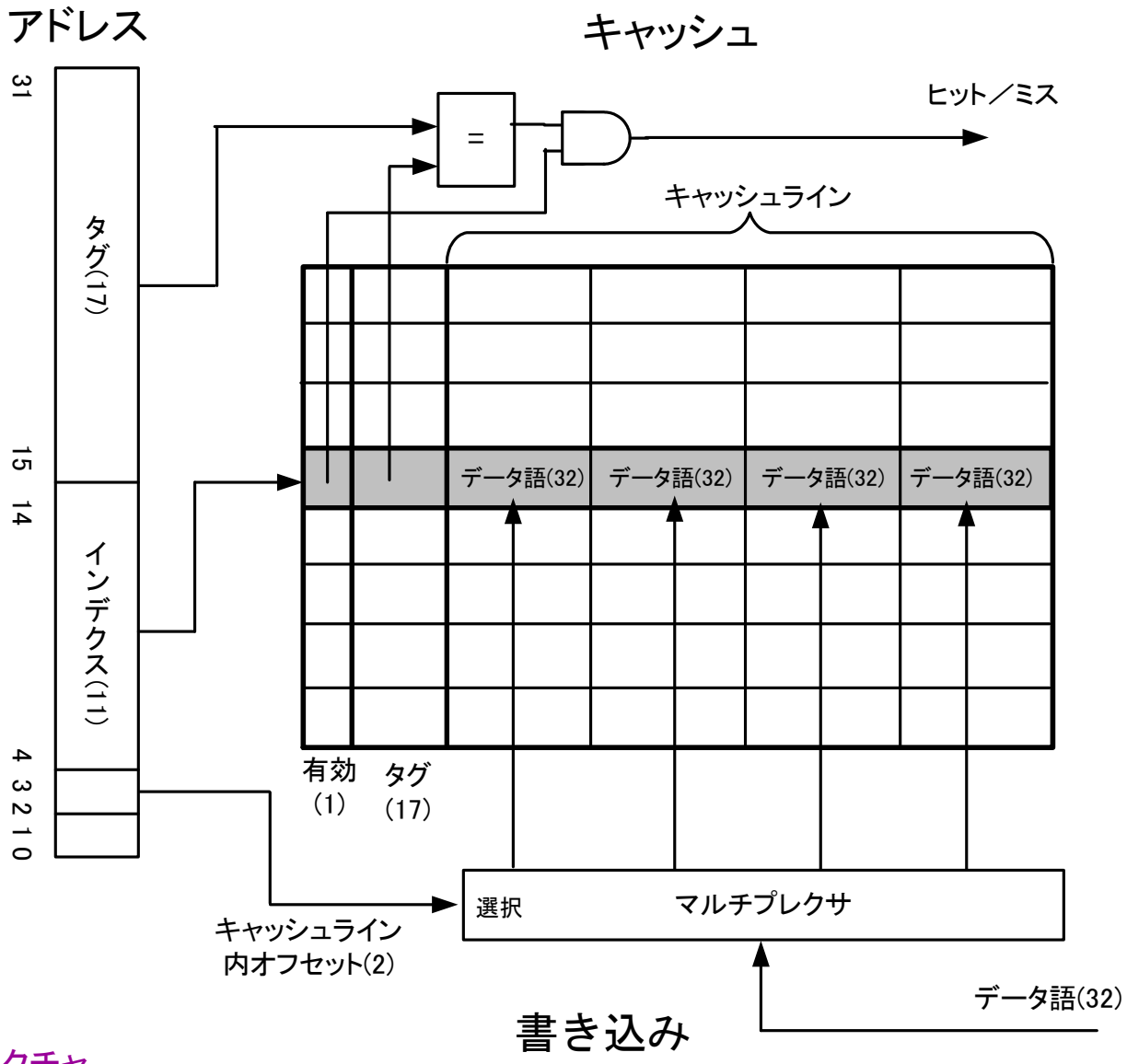
ブロックアドレス

ダイレクトマップ型キャッシュ（1）



読み出し

ダイレクトマップ型キャッシュ（2）



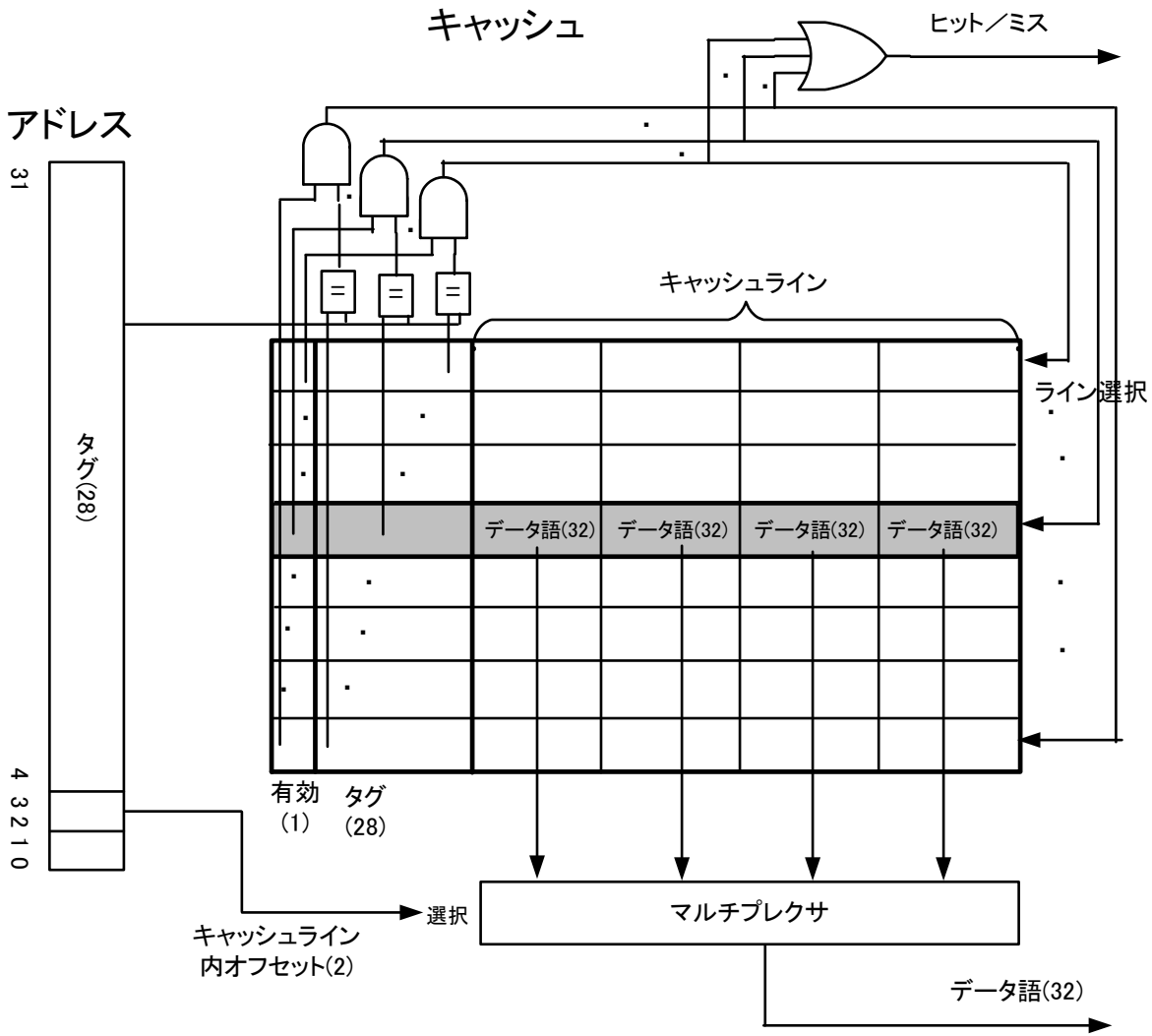
キャッシュミス

キャッシュミス　＝　３つのＣ

①初期参照ミス(compulsory miss, cold start miss)
　キャッシュラインを最初にアクセスするときに起こるミス
②競合性ミス(conflict miss, collision miss)
　同じインデクスをもつ異なるキャッシュラインにアクセスすることで起こるミス
③容量性ミス(capacity miss)
　キャッシュしたいライン数がキャッシュ容量を上回ることで起こるミス

メモ　競合性ミスはフルアソシアティブ型キャッシュでは起こらない

フルアソシアティブ型キャッシュ



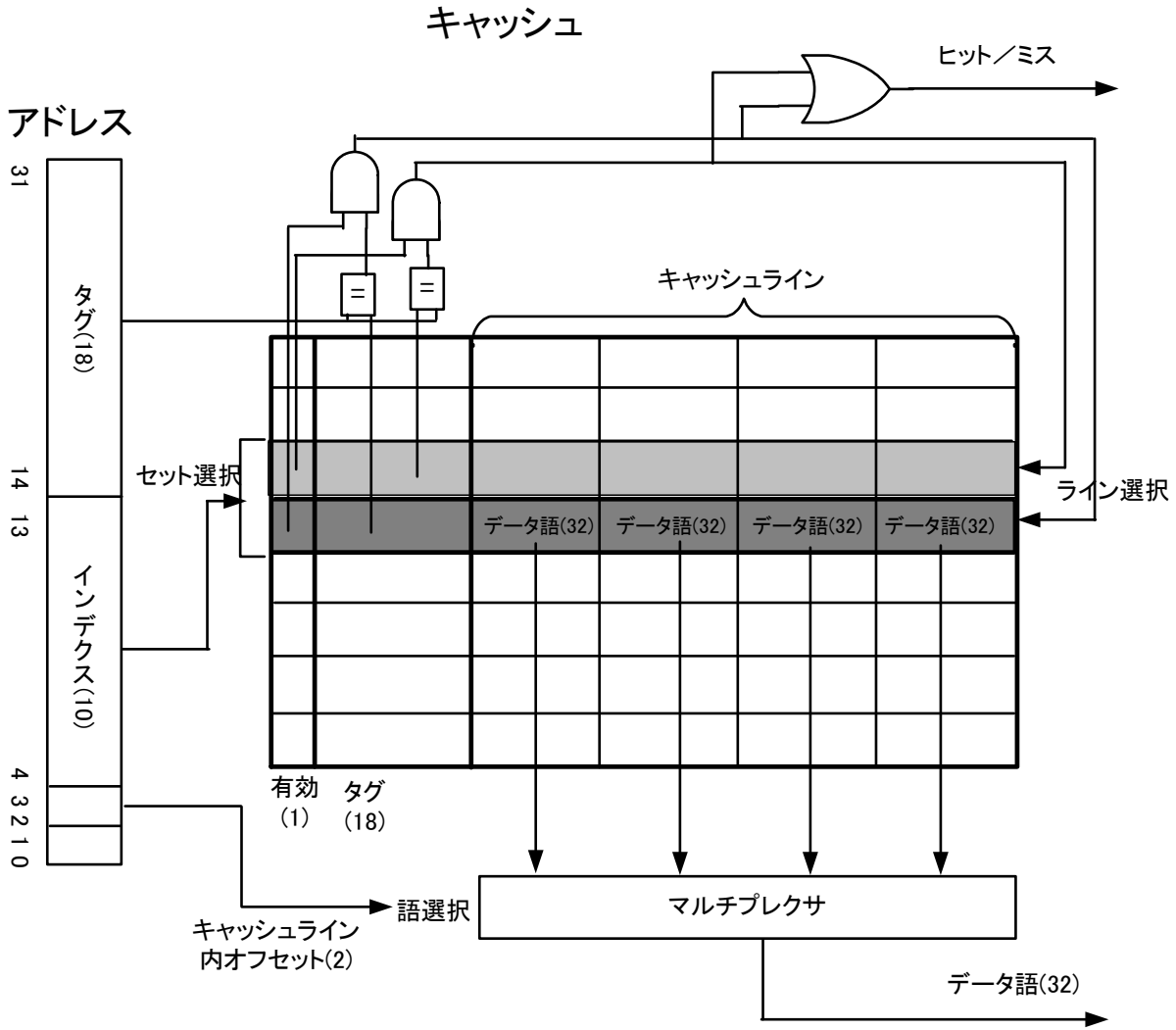
$L = S \times A$

L：ライン数
S：セット数
A：連想度

◆フルアソシアティブ型
$A = L$

◆ダイレクトマップ型
$A = 1$

セットアソシアティブ型



キャッシュ方式の比較

表 5.2　キャッシュの3つの型

	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	×	○	◎

- キャッシュラインの交換

  追い出すラインをどう決めるか？

  - ランダム
  - LRU （Least Recently Used）

    使われていない時間が最も長い(Least Recent Used)ラインを追い出しの対象とする

メモ　キャッシュの置き換えとキャッシュのタイプ

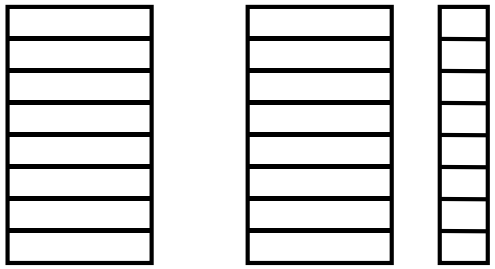
キャッシュミス時に、どのブロックを置き換えるか

- ダイレクトマップ方式
  - 置き換えるブロックは一意的に決定

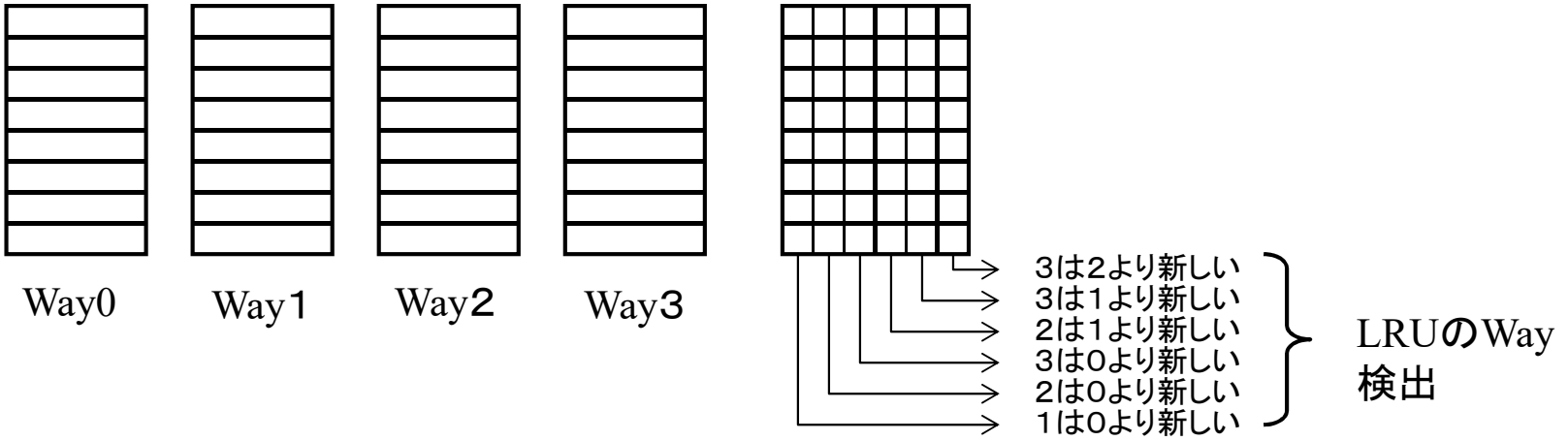
- セットアソシアティブ方式
  - A（連想数）の候補から選択
    - Random方式
    - LRU方式　　キャッシュメモリでは、正確なLRUが可能

- フルアソシアティブ方式
  - Random方式が中心（正確なLRUは無理）

メモ：LRU方式の実現



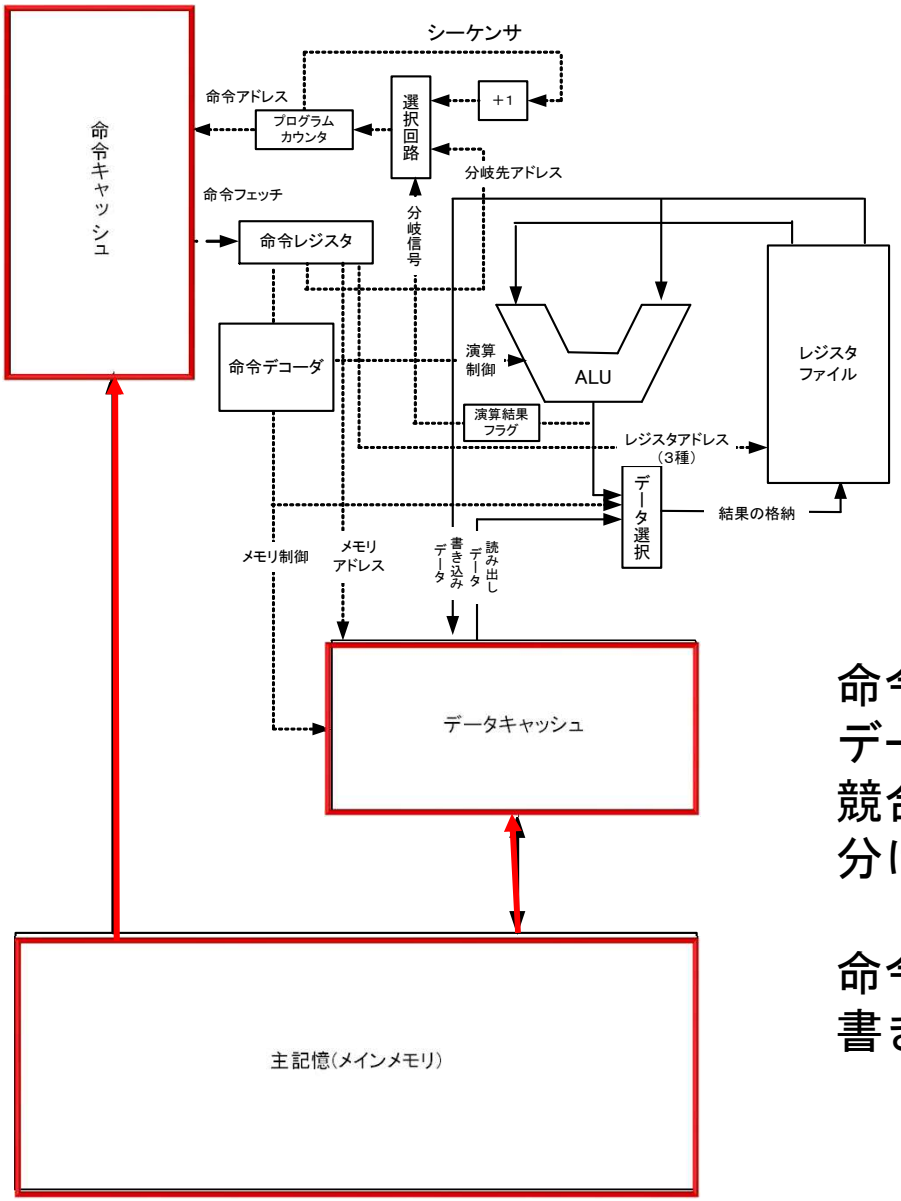
Way0にアクセスすると
1をかきこみ、Way1に
アクセスすると0を書く



ブロック置換方式の効果

	Associativity					
	Two-way		Four-way		Eight-way	
Size	LRU	Random	LRU	Random	LRU	Random
16 KB	5.18%	5.69%	4.67%	5.29%	4.39%	4.96%
64 KB	1.88%	2.01%	1.54%	1.66%	1.39%	1.53%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

キャッシュの入ったCPU



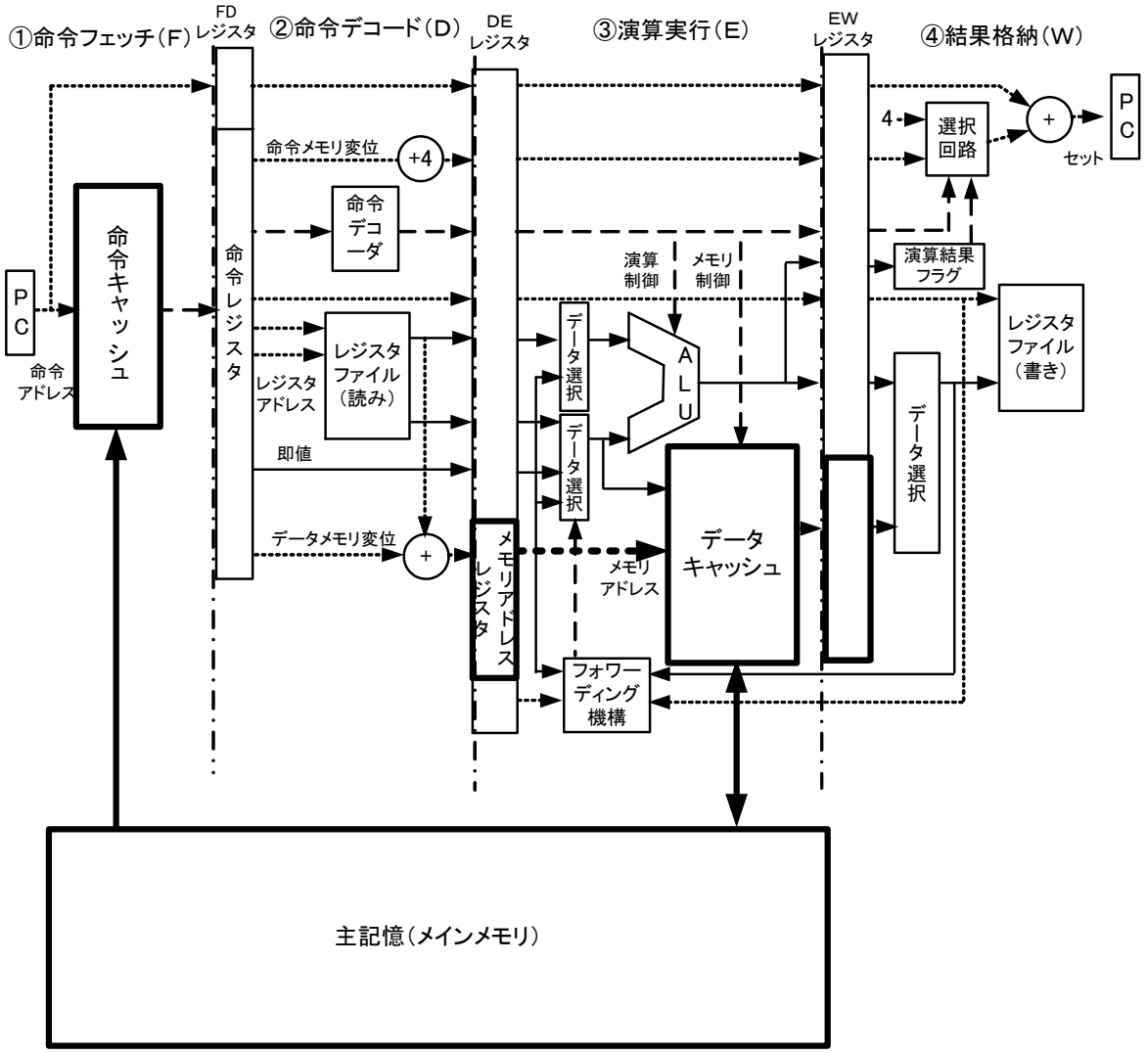
命令キャッシュと
データキャッシュは
競合を避けるために
分ける。

命令キャッシュは
書き戻し機構が不要

メモ：ID分離キャッシュと統合キャッシュのミス率

Size	Instruction cache	Data cache	Unified cache
1 KB	3.06%	24.61%	13.34%
2 KB	2.26%	20.57%	9.78%
4 KB	1.78%	15.94%	7.24%
8 KB	1.10%	10.19%	4.57%
16 KB	0.64%	6.47%	2.87%
32 KB	0.39%	4.82%	1.99%
64 KB	0.15%	3.77%	1.35%
128 KB	0.02%	2.88%	0.95%

キャッシュの入った命令パイプライン



アドレスの流れ　制御の流れ　データの流れ

キャッシュの性能

- $T_p = N \cdot (1 + r_{ls} \cdot r_{miss} \cdot t_{mstall}) / C$
  - T_p:プログラム実行時間、
  - C [Hz] :CPUのクロック速度
  - N :プログラムで実行される命令の数
  - r_{ls}:ロード・ストア命令の割合
  - r_{miss}:ロード・ストア命令ごとのキャッシュミス率
  - t_{mstall} :1回のミスによるストール時間(ミスペナルティ、miss penalty)

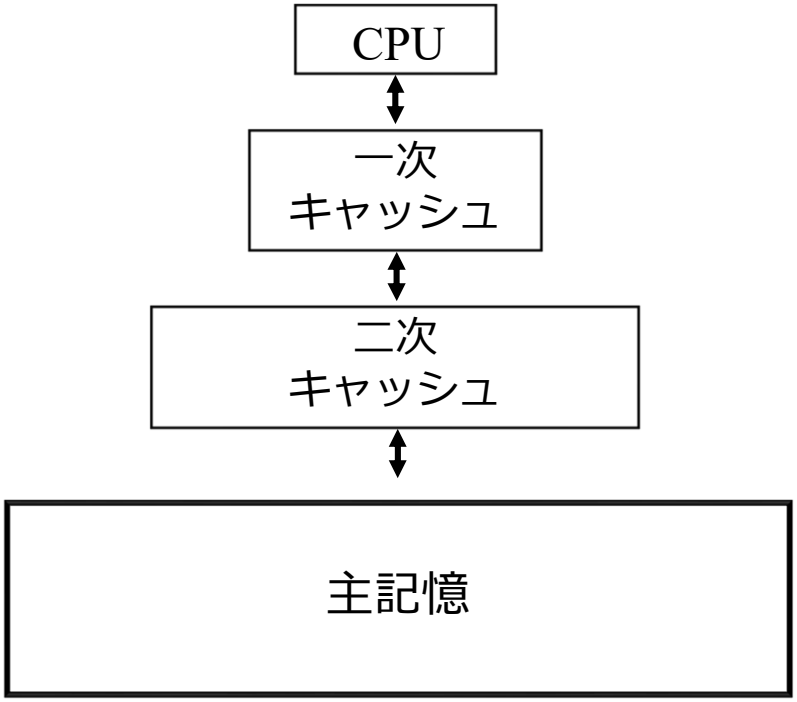
表 5.4　キャッシュミス率とミスペナルティの例（解答）

	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

$r_{ls} = 0.3$の場合

メモ

- キャッシュが複数レベルのものも存在



Intel core i7の場合
一次（1L)キャッシュ　32KB
二次（2L）キャッシュ　256KB
3次（3L)キャッシュ　8MB

問題　キャッシュの性能

- 二次キャッシュまであるCPUで
  T_p:プログラム実行時間を以下のパラメータで記述せよ

  - C [Hz] : CPUのクロック速度
  - N : プログラムで実行される命令の数
  - r_{ls}: ロード・ストア命令の割合
  - r_1: ロード・ストア命令ごとの1次キャッシュミス率
  - r_2: ロード・ストア命令ごとの2次キャッシュミス率
  - t_1 : 1回の1次キャッシュミスによるストール時間
  - t_2: 1回の2次キャッシュミスによるストール時間

回答

- Tp=N*(1+rls*r1*t1+rls*r2*t2)/C
- Rls=0.3の時の性能の比較

	R1	T1	R2	T2	相対実行時間
事例1	0				1
事例2	0.05	10	0		1.15
事例3	0.05	40	0		1.6
事例4	0.05	10	0.001	100	1.18

メモ：これはかなり理想的な場合である。
通常1次キャッシュヒット時にもオーバヘッドはある。
2次キャッシュヒット時にもオーバヘッドはある。
AMAT (Average Memory Access Time)を
AMAT = Hit TimeL1 + Miss RateL1 x (Hit TimeL2 + Miss RateL2 x Miss PenaltyL2)