

応用情報技術者試験

テクノロジ系Ⅲ

2-1-2

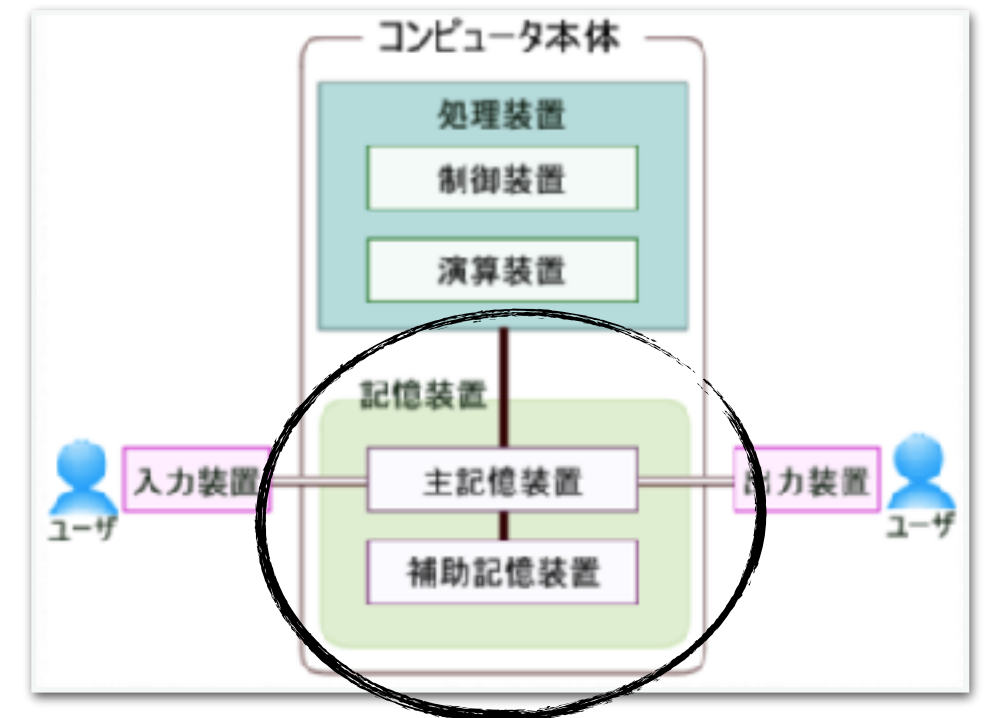
メモリ

優先度： 中



記憶装置(メモリ)とは

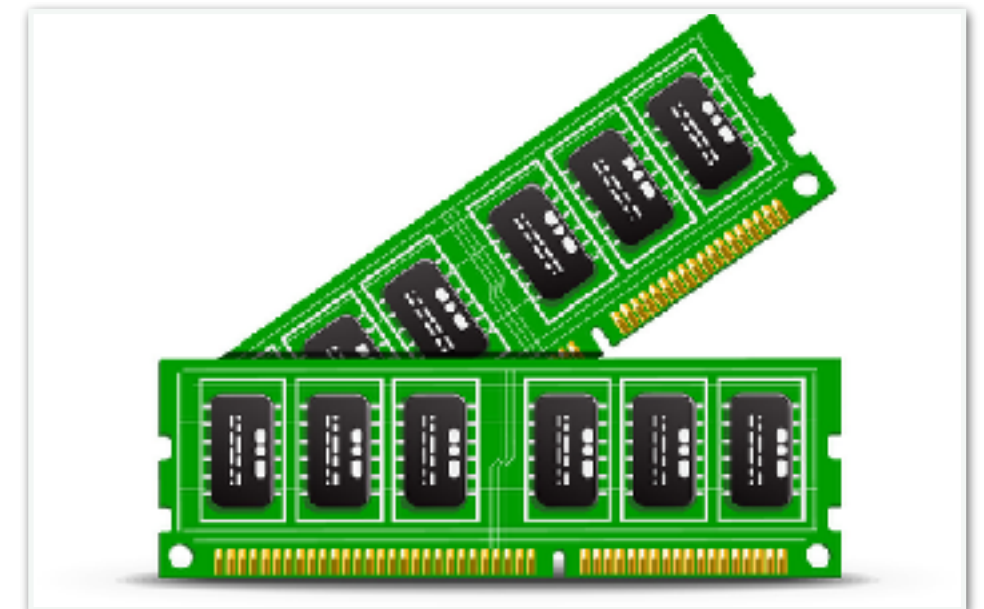
- コンピュータにおいて情報の記憶を行う装置
 - * 主記憶装置と補助記憶装置の2種類がある



RAM



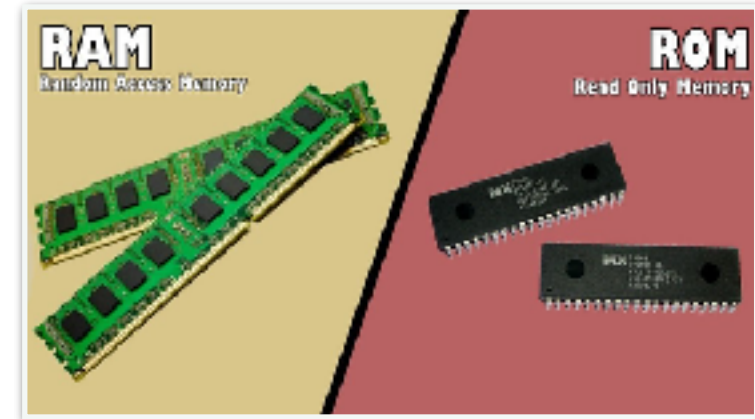
ROM



主記憶装置の種類

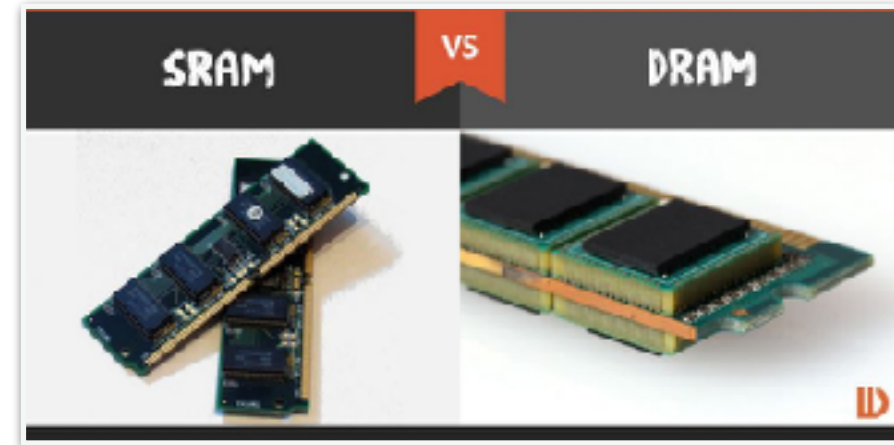


RAMとROM



- **RAM**(Random Access Memory)
 - * コンピュータが処理で使う一時的な記憶域, いわゆる「メモリー」. **揮発性**
- **ROM**(Read Only Memory)
 - * データが書き込まれ常に保存されている, 書き込みができる. **不揮発性**

RAM



- 半導体メモリを使用

- * **電源の供給がなくなると内容が消えてしまう**

- ▶ 電源を切った後も保存したい情報は補助記憶装置に退避させておき, 必要に応じてメモリに呼び出す

- RAMの種類

- * **DRAM**(Dynamic RAM): 一定時間経つとデータが消失してしまう

- * **SRAM**(Static RAM): 電源を切らない限り内容を保持している

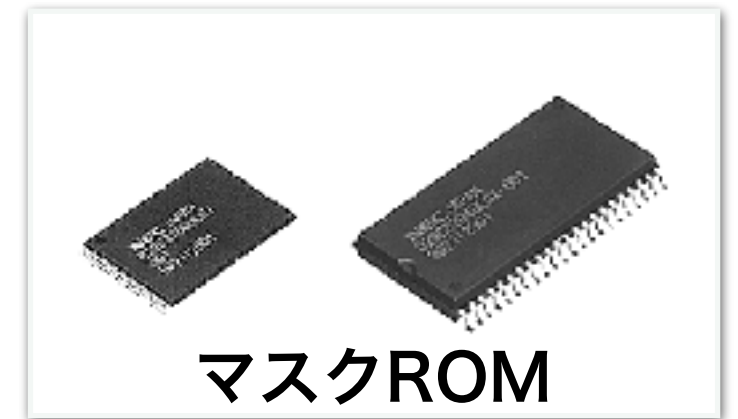
DRAMとSRAMの特徴

特徴	DRAM	SRAM
リフレッシュ	必要	不要
速度	低速	高速
電力消費	高消費電力	低消費電力
コスト	安価	高価
容量	大容量	小容量
用途	メモリ	キャッシュメモリ

応用情報技術者試験はDRAMとSRAMを比較する問題がよく出題されます！！！！

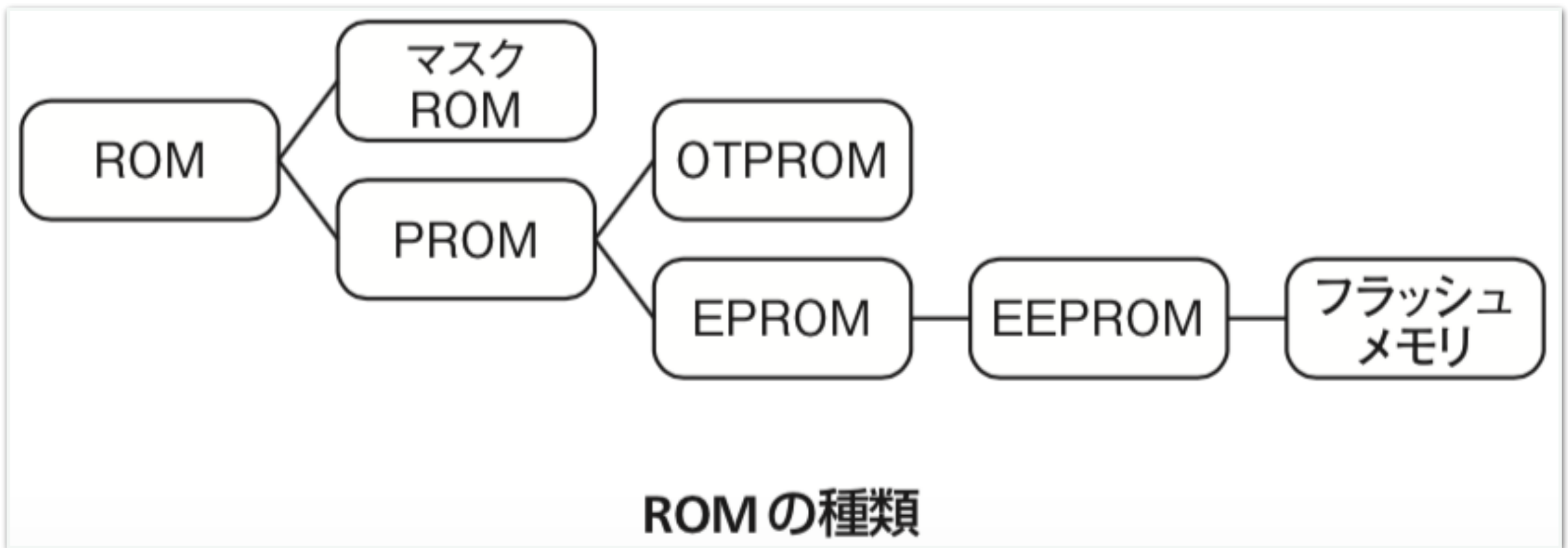
特徴の違いを理解しておきましょう！！！！

ROM



- 基本的に**読み出し専用**の記憶装置
 - * 種類によっては全消去, 書き込み, 追記が可能なことがある
- **マスクROM**: 書き換えが不可能なROM
- **PROM**(Programmable ROM): 書き込みが可能なROM
 - * **OTPROM**(One Time PROM): 1 回だけ書き込み可能
 - * **EPROM**(Erasable PROM): 書き込みと消去が可能
 - ▶ **EEPROM**(Electrically EPROM): 電圧をかけることで何度も書込みや消去が可能
 - **フラッシュメモリ**: ブロック単位での消去や書き込みを行う

ROMの種類



記憶階層



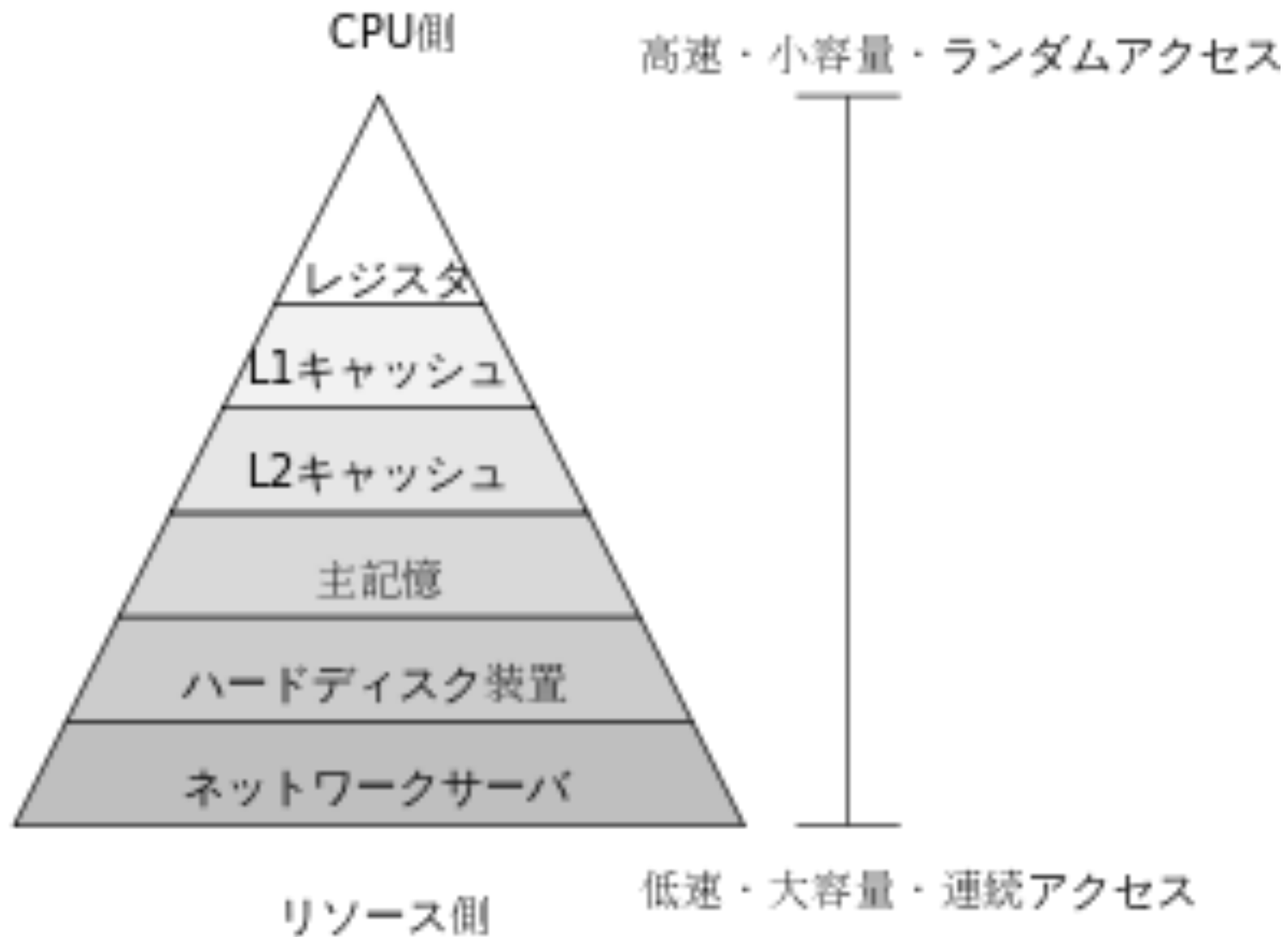
CPUと各装置へのアクセス

- 補助記憶装置に記録されているプログラムやデータを主記憶装置に読み込む
 - * CPUはそのプログラムの命令を主記憶装置から順に取り出して実行
 - ▶ 命令実行時にはCPUと主記憶装置との間でデータの読み書きが行われる
- CPUの性能が高くても、主記憶装置や補助記憶装置へのアクセス速度が遅ければ、処理の高速化は期待できない

記憶の階層化

- アクセスの速い記憶装置
 - * 順に, レジスタ, キャッシュメモリ, 主記憶装置, ディスクキャッシュ, 補助記憶装置
 - ▶ 記憶装置は高速なものほど容量が小さく高価
- CPUの理想は**高速かつ大容量**なもの
 - * 各記憶装置をうまく階層化して, CPUからはあたかも高速かつ大容量の記憶装置があるかのように見せる
 - * これを**記憶階層**という

記憶装置



記憶装置

- レジスタ
 - * CPU内にある高速アクセスができる記憶装置.
- キャッシュメモリ
 - * CPUの処理速度と主記憶へのアクセス速度の差を埋めるための, 主記憶より高速にアクセスができる装置
- ディスクキャッシュ
 - * 磁気ディスク装置より高速にアクセスができる記憶装置

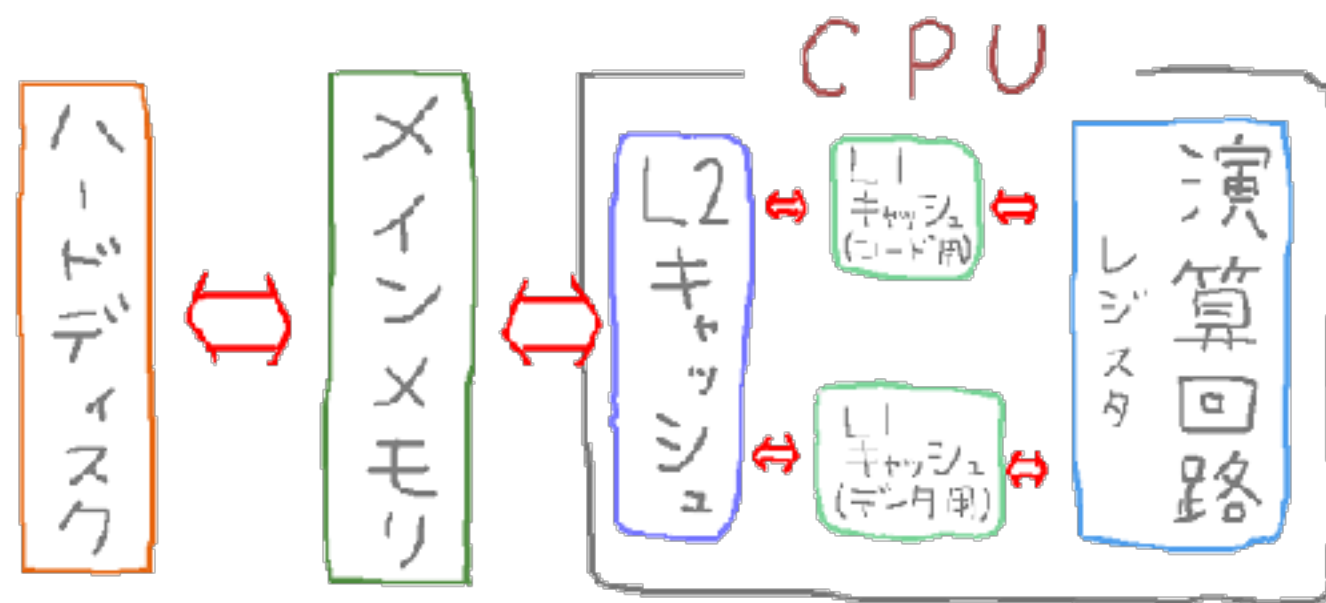
記憶装置を階層化する理由

- **局所参照性**と呼ばれる現象が根底にあるため
 - * 現在のコンピュータプログラムは全体の容量が非常に大きいため、必要なデータは到底キャッシュメモリに収まるサイズではない
 - ▶ 結局補助記憶装置にアクセスしなければならず、全体のアクセス時間が遅くなっているようなイメージを持ちがち
- しかし実際はキャッシュメモリを導入することで、実効アクセス時間を向上させることができる
 - * **時間の局所性**: 1つのデータがアクセスされた時、そのデータが近い将来再びアクセスされる可能性が高い
 - * **空間の局所性**: 連続してアクセスされるデータは隣接したアドレス空間に存在している可能性が高い

記憶装置を階層化する理由

- したがって、キャッシュメモリに取り込むデータでさえ適切であれば、ほとんどの場合キャッシュメモリを参照するだけで必要なデータが得られてしまう
- * 近年、「次にどのデータが読まれそうか」を判断する先読みアルゴリズムの急速な発達により、必要なデータがキャッシュメモリに存在する確率(**ヒット率**)は極めて高くなる

キャッシュメモリ



キャッシュメモリ

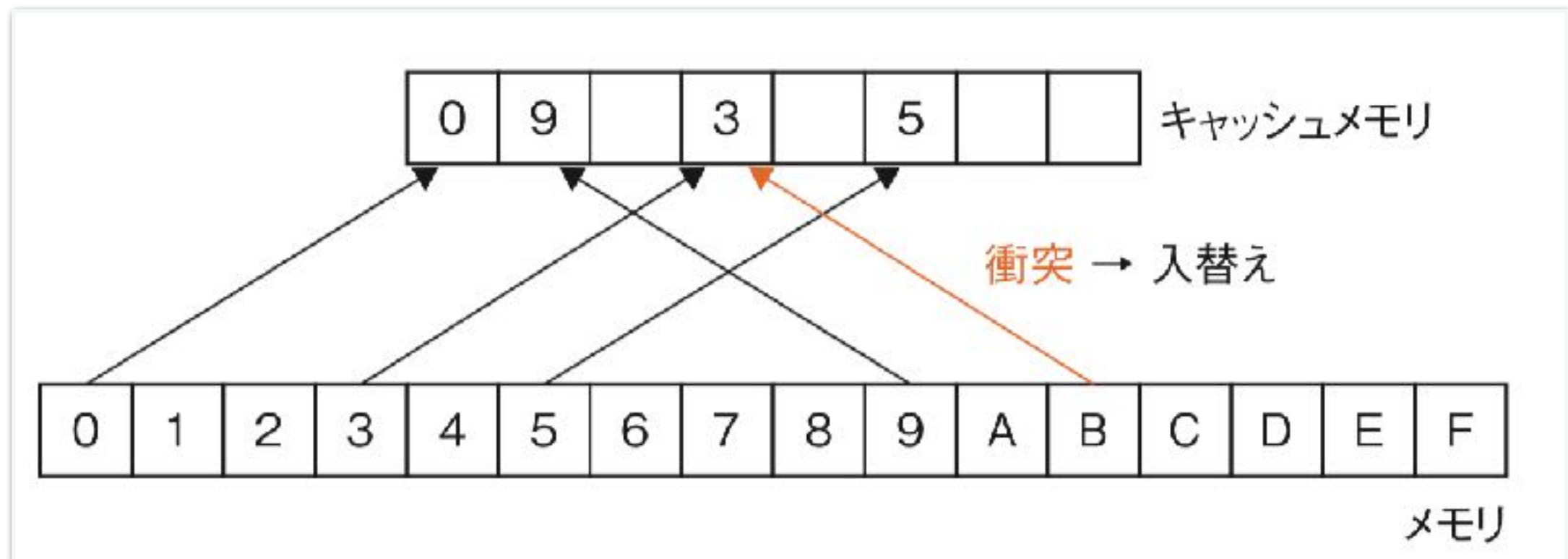
- プロセッサとメモリの性能差を埋めるために両者の間で用いられるメモリ
 - * 高速である必要があるため, **SRAM**が用いられる
 - ▶ 近年はCPU内に取り込まれ内蔵されていることが一般的
- データの格納方法や更新方式に様々なアーキテクチャを採用
 - * その効果は**ヒット率**によって変わる
- 最近のキャッシュメモリは段構造にして, CPUに近いものから一次キャッシュ, 二次キャッシュとするものが多く見られる

キャッシュメモリの データ格納方法

- ブロックと呼ばれる一定長の単位にまとめる
 - * メモリのデータがキャッシュメモリのどの部分にあるのかを管理する方法は3通りある
 - * **ダイレクトマップ方式**
 - * **フルアソシアティブ方式**
 - * **セッタソシアティブ方式**

ダイレクトマッピング方式

- メモリのアドレスごとに、キャッシュメモリの格納場所が1つに決まる方式
 - * アドレスさえわかれば場所が特定できるので検索は容易
 - ▶ その分データの衝突が起こりやすくなり、ヒット率が下がる

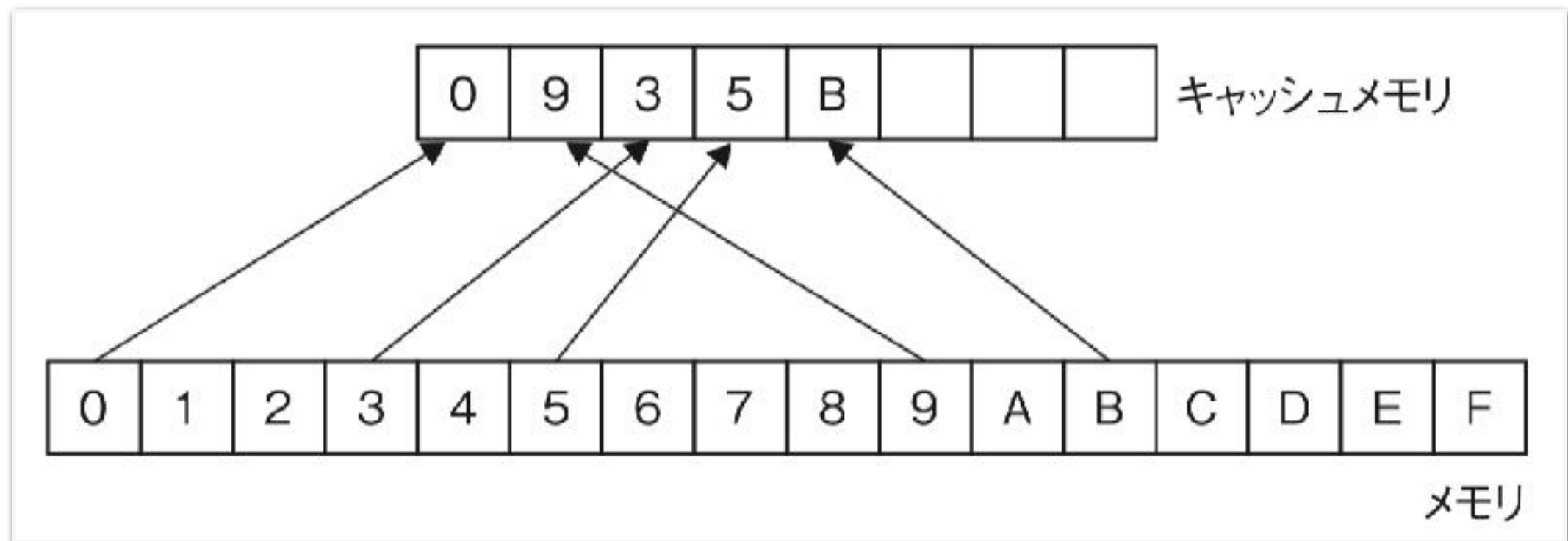


フルアソシアティブ方式

- アドレスによる振り分けを行わず, キャッシュメモリの空いているブロックならどこでも使える方式

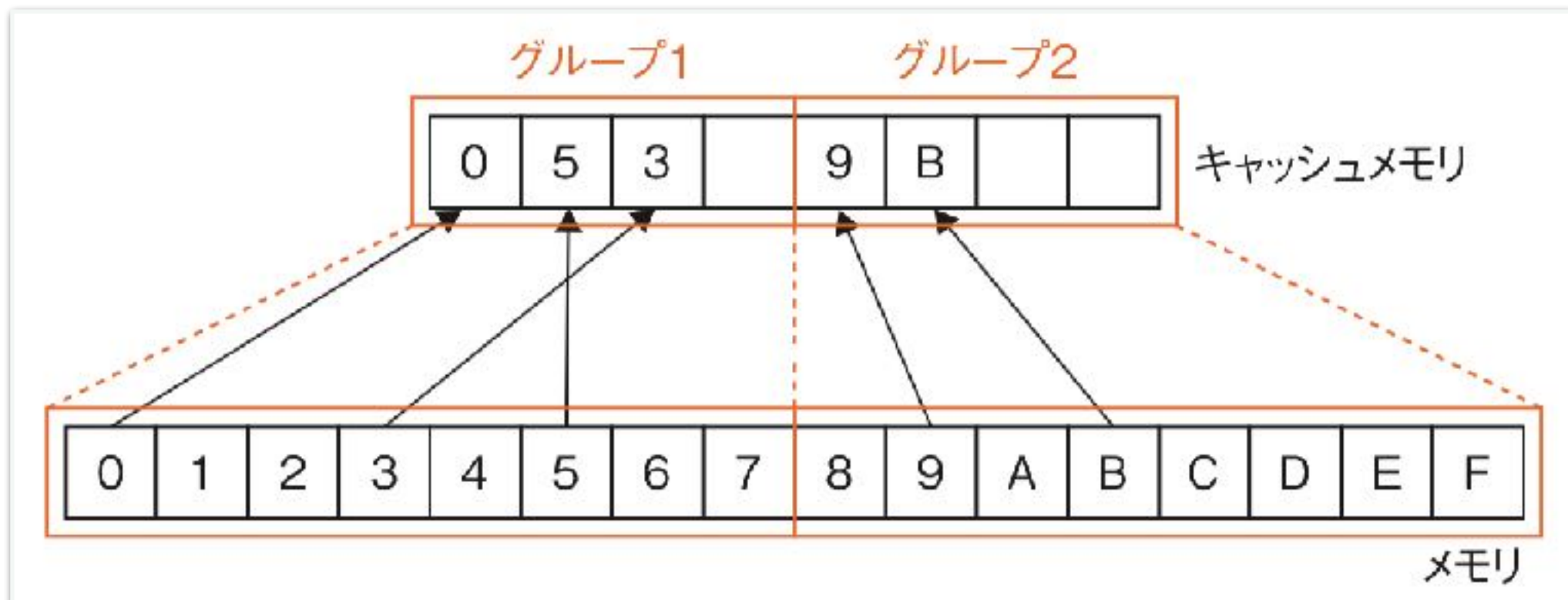
* キャッシュメモリがいっぱいになるまではデータの衝突は起こらない

- ▶ データの使用時毎回すべてのブロックを検索する必要があるため, 応答速度に問題が出てくる



セットアソシアティブ方式

- キャッシュメモリを複数のグループに分け, そのグループ内でならどこでも使える
 - * セットアソシアティブ方式とフルアソシアティブ方式では, 連想メモリ (CAM: Content Addressable Memory) を使用することで検索を高速化している

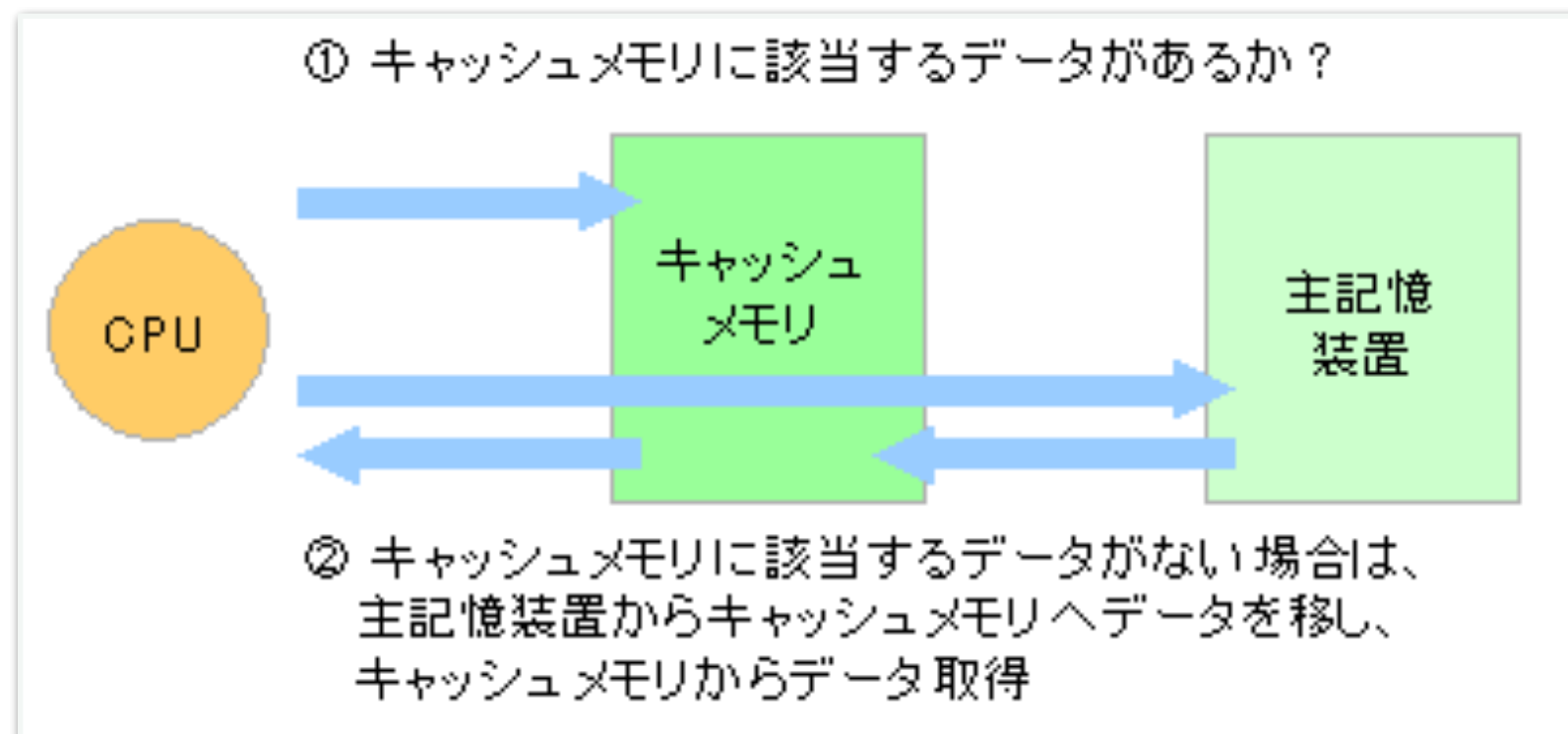


主記憶の実効アクセス 時間



主記憶のアクセスにかかる時間

- キャッシュメモリは主記憶よりも高速にアクセスできる
 - * しかしキャッシュメモリにはCPUがこれから利用すると予想されるデータしか記憶されていない
 - ▶ キャッシュメモリ上にCPUが利用するデータがある場合には高速にアクセスできるが、なければ主記憶にアクセスする



実効アクセス時間

- **ヒット率**: 利用したいデータがキャッシュメモリに存在する確率
 - * 逆に利用したいデータがキャッシュに存在しないことを**ミスヒット**という

- **実効アクセス時間の求め方**

$$\underline{TE = TC * P + TM * (1 - P)}$$

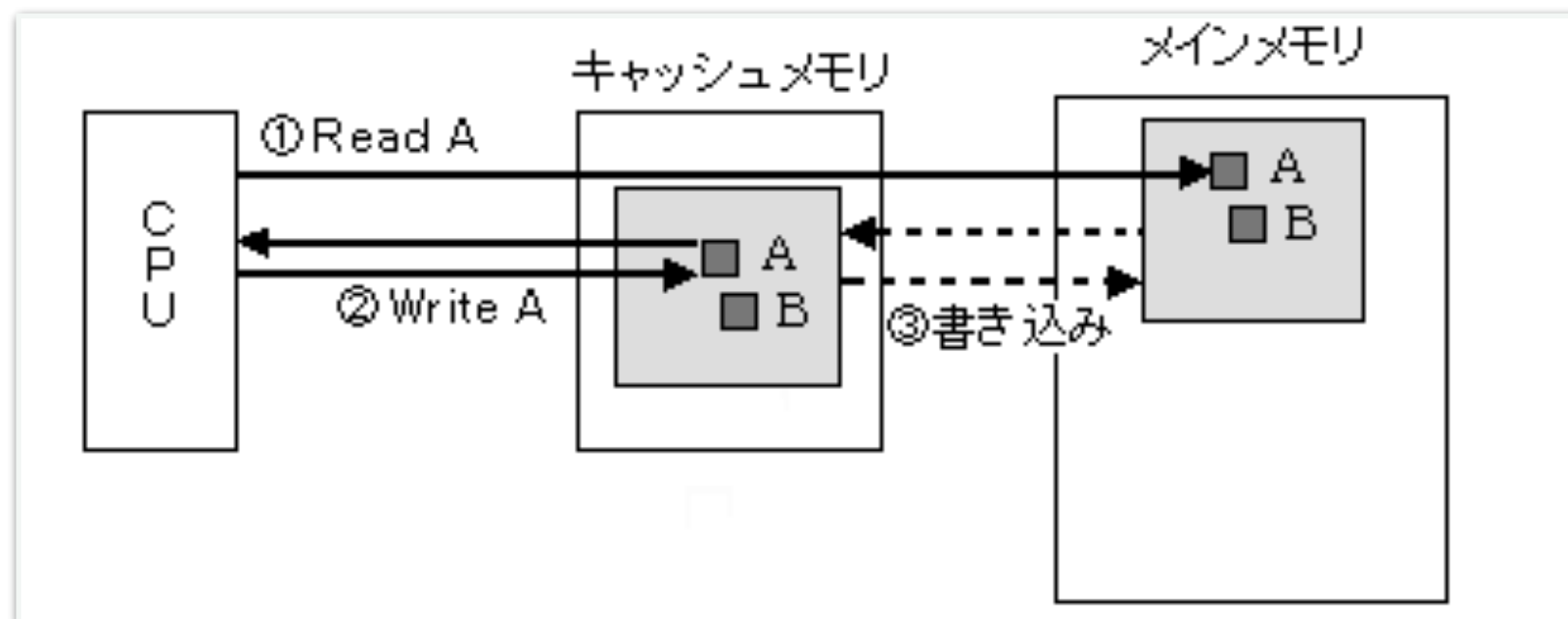
- * **TE**: 実効アクセス時間
- * **TC**: キャッシュメモリのアクセス時間
- * **P**: ヒット率
- * **TM**: 主記憶のアクセス時間

主記憶への書込み方式

- キャッシュメモリ上のデータはいずれ主記憶に書き出す必要がある
 - * 書き出しのタイミングにより2つの方法がある
 - ▶ **ライトスルー方式**
 - ▶ **ライトバック方式**

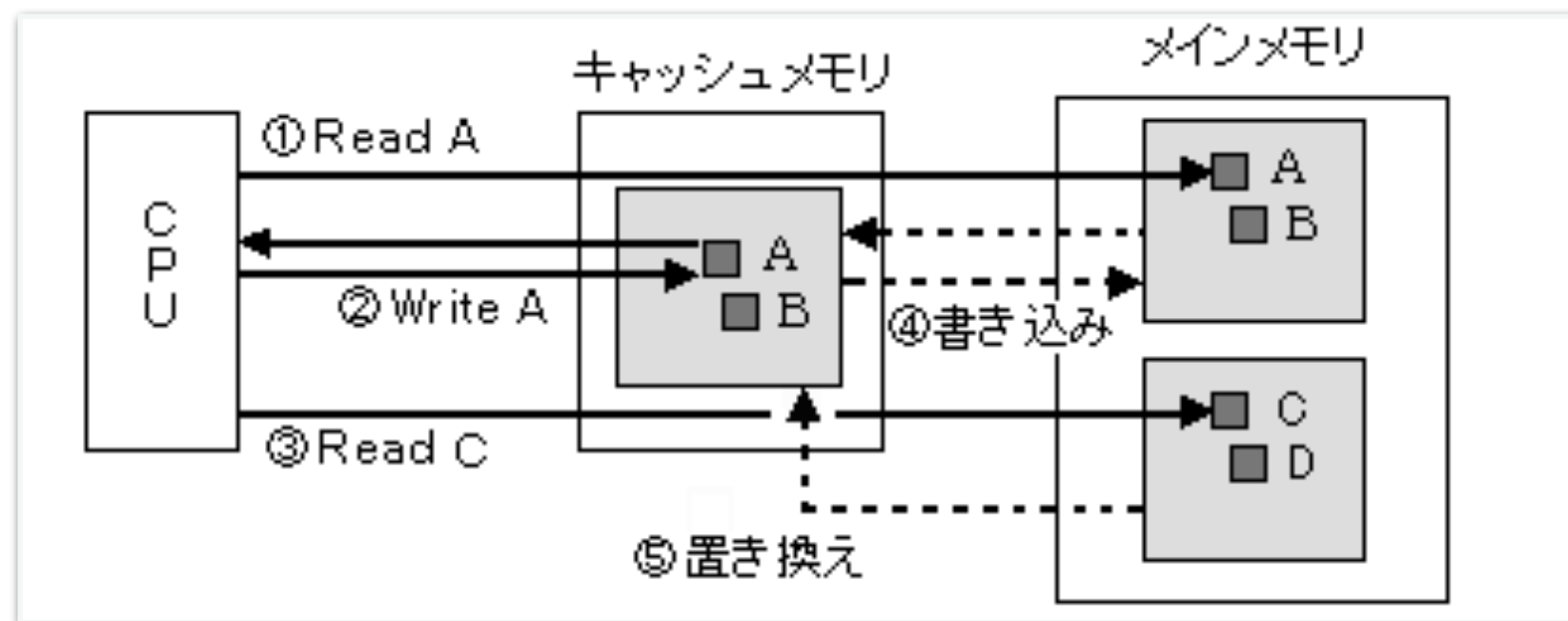
ライトスルー方式

- CPUからデータの書き込み命令が発生した時, キャッシュメモリと同時に主記憶にも書き込みを行う方式
 - * 両者の内容が必ず一致するのでデータの一貫性(**コヒーレンシ**)は保持できる
 - ▶ デメリット: 主記憶への書き出しが終わるまでCPUは別の処理に移行できない



ライトバック方式

- CPUからデータの書き込み命令が発生した時, キャッシュメモリにだけデータを書き込んでおき, 主記憶への書き込みはキャッシュメモリからそのデータが追い出された時に行う方式
 - * CPUは主記憶へ書き込む時間を待たなくてすむためライトスルー方式に比べて高速性が得られる
 - ▶ デメリット: 主記憶への書き込み(キャッシュメモリ上の更新されたデータの主記憶への反映)は後になるので, キャッシュメモリと主記憶間のデータの一貫性(コヒーレンシ)を保持できない



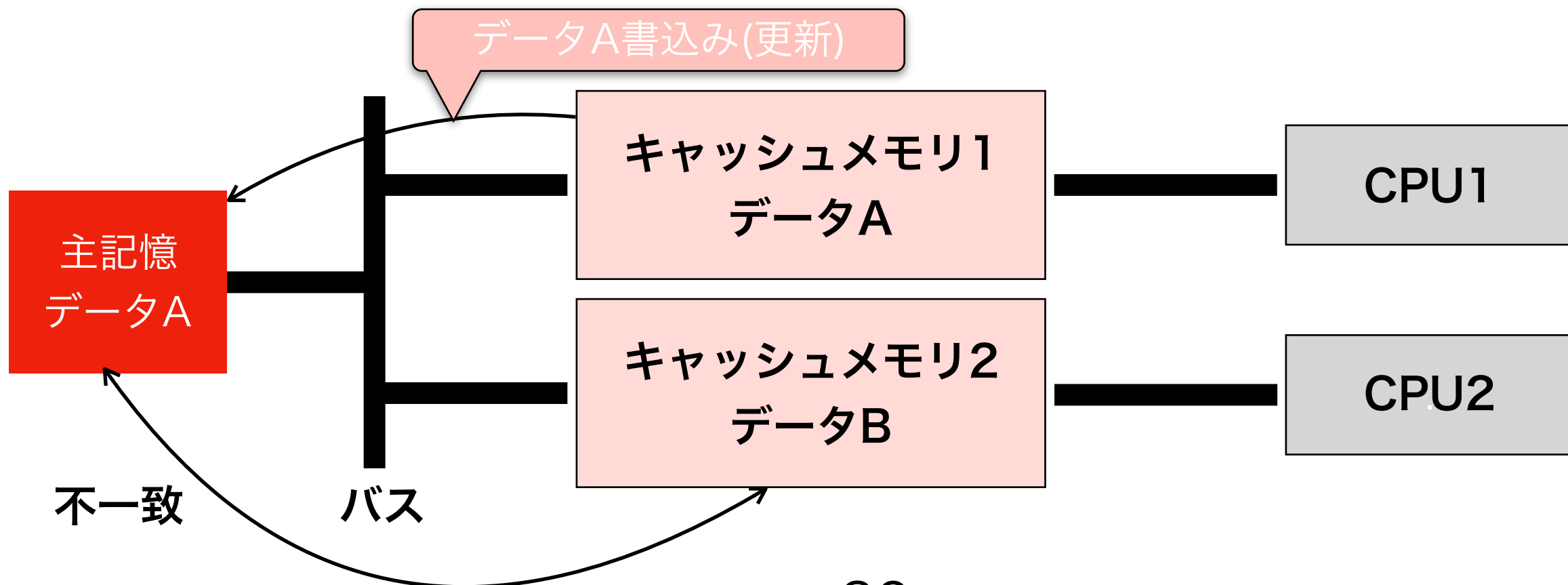
LRUアルゴリズム

- ライトバック方式を有効に機能させるには、**キャッシュメモリ上のどのデータをどのタイミングで主記憶に追い出す(書き出す)のか**、という点が重要になる
 - * どのデータを追い出すのかタイミングを誤れば、今追い出したばかりのデータがまた必要になり、すぐに主記憶から読み直さなければならず、これを繰り返すとヒット率が下がる
- 効率的にデータを追い出すため、最も利用されるのが**LRU**(Least Recently Used)アルゴリズム
 - * ここ最近で最も長い時間利用されていないものを追い出しの対象とするアルゴリズム
 - * 局所参照性から、最近使われていないものは将来的にも使われないという推測が成立するため、効率的だと言える

マルチプロセッサにおける

データ整合性

- 主記憶を共有するマルチプロセッサシステムで同じデータを各々のキャッシュメモリに保持している場合, 他方により主記憶のデータが更新されると, 自身のキャッシュメモリ上のデータと主記憶のデータに不一致が生じる



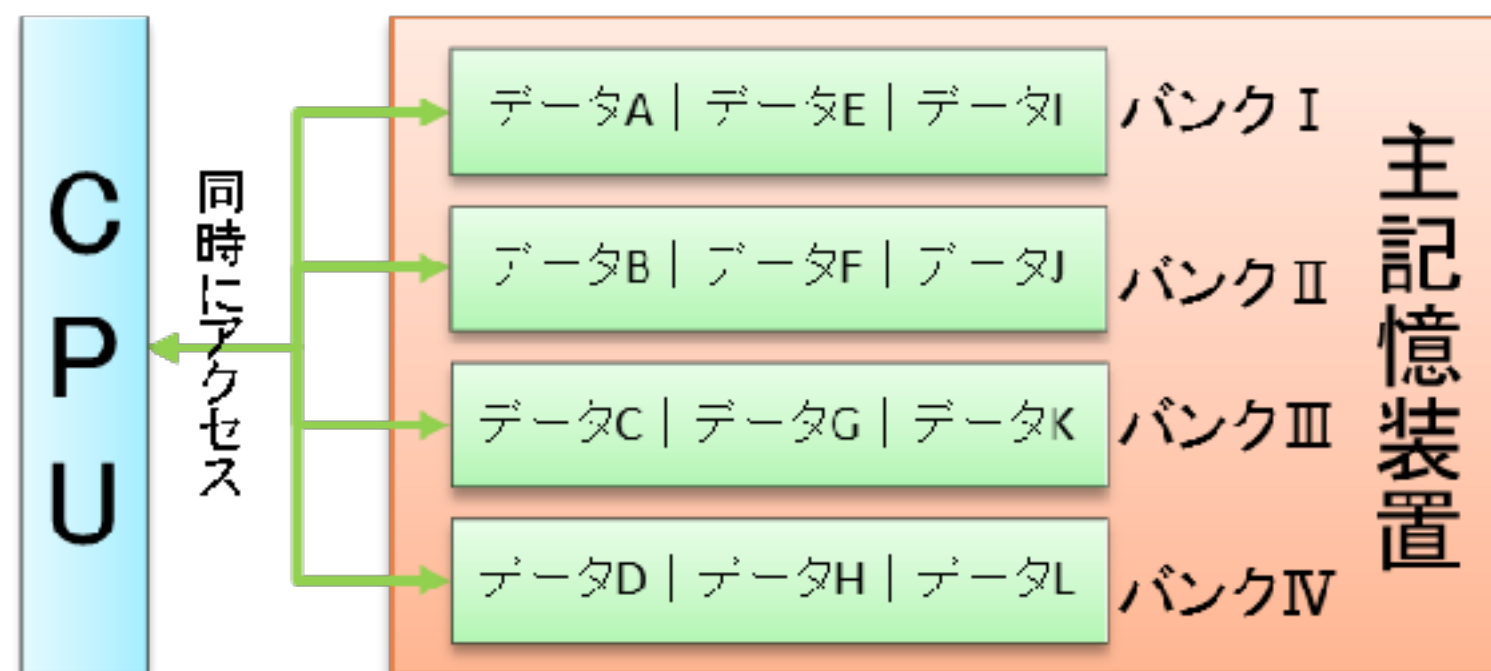
マルチプロセッサにおける

データ整合性

- **スヌープ方式**: 各キャッシュメモリの内容を正しく保つための方式
 - * 共有する主記憶のデータが変更されたかどうかをバスを介して監視し(この動作を**バススヌープ**という), 自信に影響を及ぼす変更があった場合, 自身のもつ当該データを最新データで更新するかあるいは無効にする
- ライトバック方式の場合は, 主記憶に最新のデータが存在しない可能性があるため, 主記憶上の古いデータを読み込む危険性がある
 - * スヌープ方式ではこれを回避するため, 他のキャッシュと更新情報を交換することで, どのキャッシュに最新のデータが存在するかを知ることができる

メモリアンタリーブ

- 主記憶へのアクセスを擬似的に高速化することでCPUを待ち状態にしないための方法
 - * メモリアンタリーブでは, 独立にアクセスできる複数のメモリバンク(主記憶をいくつかのアクセス単位に分割したもの)を用意し, これを並行してアクセスすることで見かけ上のアクセス速度を向上させることができる
 - ▶ しかし連続したアドレスへのアクセスでないと高速化の効果が薄くなる



それでは過去問を解い
てみましょう！



問1

メモリアンタリーブの説明として、適切なものはどれか。

出典：平成30年春期 問11

- ア 主記憶と外部記憶を一元的にアドレス付けし、主記憶の物理容量を超えるメモリ空間を提供する。
- イ 主記憶と磁気ディスク装置との間にバッファメモリを置いて、双方のアクセス速度の差を補う。
- ウ 主記憶と入出力装置との間でCPUとは独立にデータ転送を行う。
- エ 主記憶を複数のバンクに分けて、CPUからのアクセス要求を並列に処理できるようにする。

解答

正解

エ

解説

メモリアンタリーブは、メモリアクセス高速化のための技法で、物理的にひとつの主記憶を同時アクセス可能な複数の論理的な領域(バンク)に分け、これに並列アクセスすることで見かけ上のアクセス時間を短縮することができます。

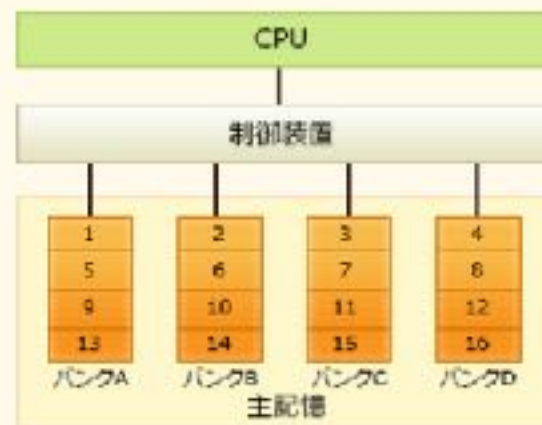


図 メモリアンタリーブ

ア：“主記憶と外部記憶を一元的にアドレス付けし、主記憶の物理容量を超えるメモリ空間を提供する。”

仮想記憶の説明です。

イ：“主記憶と磁気ディスク装置との間にバッファメモリを置いて、双方のアクセス速度の差を補う。”

ディスクバッファの説明です。

ウ：“主記憶と入出力装置との間でCPUとは独立にデータ転送を行う。”

DMA制御方式の説明です。

エ：“主記憶を複数のバンクに分けて、CPUからのアクセス要求を並列に処理できるようにする。”
正しい。メモリアンタリーブの説明です。

問2

容量が a Mバイトでアクセス時間が x ナノ秒の命令キャッシュと、容量が b Mバイトでアクセス時間が y ナノ秒の主記憶をもつシステムにおいて、CPUからみた、主記憶と命令キャッシュとを合わせた平均アクセス時間を表す式はどれか。ここで、読み込みたい命令コードがキャッシュに存在しない確率を r とし、キャッシュメモリ管理に関するオーバーヘッドは無視できるものとする。

出典：平成29年秋期 問11

ア $\frac{(1-r) \cdot a}{a+b} \cdot x + \frac{r \cdot b}{a+b} \cdot y$

イ $(1-r) \cdot x + r \cdot y$

ウ $\frac{r \cdot a}{a+b} \cdot x + \frac{(1-r) \cdot b}{a+b} \cdot y$

エ $r \cdot x + (1-r) \cdot y$

解答

正解

イ

解説

メモリの平均アクセス時間に関する問題は頻出です。この種の問題はメモリの容量を問題文にいれたり、存在しない確率を r としてみたり、受験者を戸惑わせようとする目論見が見えますが、主記憶の平均アクセス時間の求め方は、

キャッシュメモリのアクセス時間 $\times$ ヒット率 $+$ 主記憶のアクセス時間 $\times$ (1-ヒット率)

なので、メモリの容量は関係ありません。ここで立式に係る値は次の4つです。

- キャッシュメモリのアクセス時間 $=x$
- ヒット率 $=$ 存在しない確率の逆なので $(1-r)$
- 主記憶のアクセス時間 $=y$
- ヒットしない確率 $=r$

これを公式に代入すると「 $(1-r) \cdot x + r \cdot y$ 」になります。したがって「イ」が正解です。

一般にキャッシュメモリ容量が大きくなればヒット率は上昇するはずです。したがってヒット率を計算するときにはキャッシュメモリと主記憶の容量が計算要素として使われることとなります。しかしこの問題の場合は、ヒット率が定数として問題文の条件に示されているので、各メモリ容量を計算式に含める必要はない訳です。

問3

SRAMと比較した場合のDRAMの特徴はどれか。

出典：平成29年秋期 問20

- ☐ ア 主にキャッシュメモリとして使用される。
- ☐ イ データを保持するためのリフレッシュ又はアクセス動作が不要である。
- ☐ ウ メモリセル構成が単純なので、ビット当たりの単価が安くなる。
- ☐ エ メモリセルにフリップフロップを用いてデータを保存する。

解答

|| 正解

ウ

|| 解説

DRAM(Dynamic Random Access Memory)は、コンデンサに電荷を蓄えることにより情報を記憶し、電源供給が無くなると記憶情報も失われる揮発性メモリです。集積度を上げることが比較的簡単にできるためコンピュータの主記憶装置として使用されています。

ア：“主にキャッシュメモリとして使用される。”

DRAMは主記憶として使用されています。キャッシュメモリには、アクセス速度がより速いSRAMが使用されています。

イ：“データを保持するためのリフレッシュ又はアクセス動作が不要である。”

DRAMは蓄えた電荷の有無で情報を保持するコンデンサによって記憶を行います。また、アクセス時には電荷の有無の情報が破壊されるので、再書き込みの動作が行われます。そして、コンデンサに蓄えた電荷は時間がたつと失われてしまうためにリフレッシュ操作を随時行う、もしくはアクセスする必要があります。

ウ：“メモリセル構成が単純なので、ビット当たりの単価が安くなる。”

正しい。SRAMの1ビットはトランジスタが4個以上と面積が大きいですが、DRAMの1ビットはトランジスタ1個とコンデンサ1個で構成されており小面積で済みます。これによりDRAMのビット当たりの単価はSRAMよりも安くなります。

エ：“メモリセルにフリップフロップを用いてデータを保存する。”

フリップフロップを用いるのはSRAMです。

問4

キャッシュの書込み方式には、ライトスルー方式とライトバック方式がある。ライトバック方式を使用する目的として、適切なものはどれか。

出典：平成26年秋期 問9

- ア キャッシュと主記憶の一貫性(コヒーレンシ)を保ちながら、書込みを行う。
- イ キャッシュミスが発生したときに、キャッシュの内容の主記憶への書き戻しを不要にする。
- ウ 個々のプロセッサがそれぞれのキャッシュをもつマルチプロセッサシステムにおいて、キャッシュ管理をライトスルー方式よりも簡単な回路構成で実現する。
- エ プロセッサから主記憶への書込み頻度を減らす。

解答

正解

エ

解説

ライトスルー方式とライトバック方式には、次のような違いがあります。

ライトスルー方式(write through)

CPUから書き込む命令が出たときにキャッシュメモリと同時に主記憶にも書き込む方式。データの整合性は得られるが処理速度は低下する。

→常に主記憶とキャッシュの内容が一致するため一貫性の確保は容易だが、主記憶への書き込みが頻繁に行われるので遅い

ライトバック方式(write back)

CPUから書き込む命令が出たときにキャッシュメモリだけに書き込む方式。主記憶への書き込みはキャッシュメモリからデータが追い出されるときに行われる。

→主記憶とキャッシュの内容が一致しないため一貫性を保つための制御が複雑になるが、主記憶への書き込み回数が少ないため速い

ア：“キャッシュと主記憶の一貫性(コヒーレンシ)を保ちながら、書き込みを行う。”

ライトスルー方式を使用する目的です。ライトバック方式では、キャッシュと主記憶の内容が常に一致しているわけではありません。

イ：“キャッシュミスが発生したときに、キャッシュの内容の主記憶への書き戻しを不要にする。”

ライトスルー方式を使用する目的です。キャッシュミスが発生すると、キャッシュ内の必要のないデータを主記憶に移動し、要求されたデータを主記憶からキャッシュにコピーします。ライトバック方式では該当データがキャッシュから追い出されたタイミングで主記憶に書き出すので、一貫性を保つために書き戻し処理が必要となります。

ウ：“低々のプロセッサがそれぞれのキャッシュをもつマルチプロセッサシステムにおいて、キャッシュ管理をライトスルー方式よりも簡単な回路構成で実現する。”

ライトスルー方式を使用する目的です。ライトスルー方式と比較した場合、ライトバック方式の同期制御は複雑になります。

エ：“プロセッサから主記憶への書き込み頻度を減らす。”

正しい。ライトバック方式を使用する目的です。

問5

キャッシュメモリにおけるダイレクトマップ方式の説明として、適切なものはどれか。

出典：平成21年秋期 問10

- ア アドレスが連続した二つ以上のメモリブロックを格納するセクタを、キャッシュ内の任意のロケーションに割り当てる。
- イ 一つのメモリブロックをキャッシュ内の単一のロケーションに割り当てる。
- ウ メモリブロックをキャッシュ内の任意のロケーションに割り当てる。
- エ メモリブロックをキャッシュ内の二つ以上の配置可能なロケーションに割り当てる

解答

正解

イ

解説

ダイレクトマップ(マッピング)方式は、キャッシュメモリへの割付方式のひとつで、主記憶のアドレスにハッシュ演算を行い、その結果から対応するキャッシュメモリのアドレスを導出します。ハッシュ演算で計算するもので主記憶のアドレスとキャッシュメモリのアドレスが1対1で対応します。

一般的にダイレクトマッピングで主記憶のブロック番号から、キャッシュメモリのブロック番号を算出するハッシュ関数は、

$$(\text{主記憶のブロック番号}) \bmod (\text{キャッシュメモリのブロック数})$$

の式が使用されます。

キャッシュメモリの割付方式には、ダイレクトマッピングのほかにフルアソシアティブとセットアソシアティブがあります。

フルアソシアティブ

主記憶のブロックが、キャッシュメモリのどのブロックにも対応付けられる方式です。ハッシュ演算の結果によらずどこにも書き込めるので、キャッシュメモリの利用率が向上しますが、ダイレクトマッピングよりも計算量がかかるオーバーヘッドが増加し、システムの仕事量を複雑になってしまうデメリットもあります。

セットアソシアティブ

連続したキャッシュブロックをセットとしてまとめ、その中であればどのブロックにも格納できるようにした方式です。

ダイレクトマッピングでは、ハッシュ関数で算出されたブロックに既にデータが存在する場合には追い出しが発生しますが、セットアソシアティブでは、一つのセットに複数のブロックがあるので追い出しをしなくてもデータを格納できます。

セット内のブロック数がN個のとき、Nウェイ・セットアソシアティブと言います。既存、コンピュータでは、4ウェイまたは8ウェイが主流です。

A : “アドレスが格納した二つ以上のメモリブロックを格納するセクター。キャッシュ内の任意のロケーションに割り当てる。”

ダイレクトマッピング方式では、主記憶のアドレスとキャッシュ内の格納ブロックは一対一に対応します。任意の場所に格納するのはフルアソシアティブ方式の特徴です。

B : “一つのメモリブロックをキャッシュ内の単一のロケーションに割り当てる。”

正しい。

C : “メモリブロックをキャッシュ内の任意のロケーションに割り当てる。”

フルアソシアティブの説明です。

D : “メモリブロックをキャッシュ内の二つ以上の配置可能なロケーションに割り当てる”
セットアソシアティブの説明です。

問6

15Mバイトのプログラムを圧縮した状態でフラッシュメモリに格納している。プログラムの圧縮率が40%、フラッシュメモリから主記憶への転送速度が20Mバイト／秒であり、1Mバイトに圧縮されたデータの展開に主記憶上で0.03秒が掛かるとき、このプログラムが主記憶に展開されるまでの時間は何秒か。ここで、フラッシュメモリから主記憶への転送と圧縮データの展開は同時には行われないものとする。

出典：平成29年春期 問11

ア 0.48

イ 0.75

ウ 0.93

エ 1.20

解答

|| 正解

ア

|| 解説

15Mバイトのデータが40%に圧縮されると、圧縮後のデータ量は、

$$15 \times 0.4 = 6 \text{ (Mバイト)}$$

データは圧縮された状態でフラッシュメモリから主記憶に転送された後、主記憶上で展開されるので、まず主記憶への転送時間を求めます。圧縮されたデータが6Mバイト、転送速度が20Mバイト／秒なので、転送時間は、

$$6 \div 20 = 0.3 \text{ (秒)}$$

さらに1Mバイトの展開につき主記憶上で0.03秒が掛かるので、展開に要する時間は、

$$0.03 \times 6 = 0.18 \text{ (秒)}$$

したがって転送～展開に要する総時間は、

$$0.3 + 0.18 = \mathbf{0.48} \text{ (秒)}$$