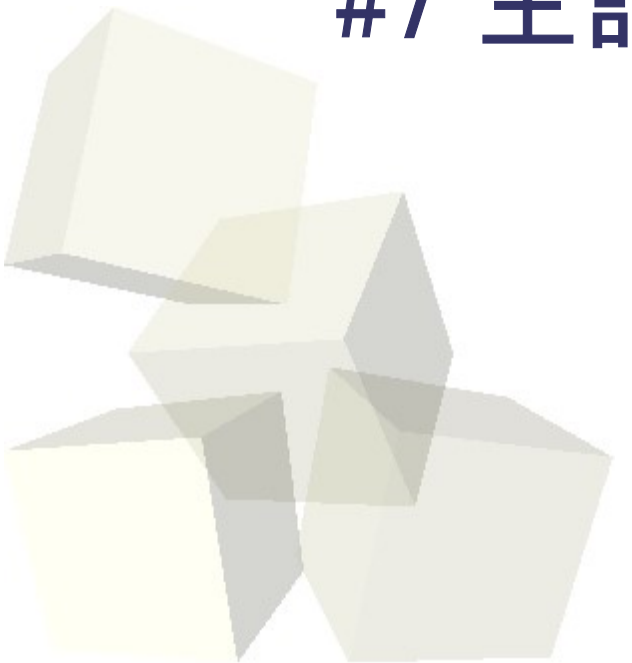
オペレーティングシステム

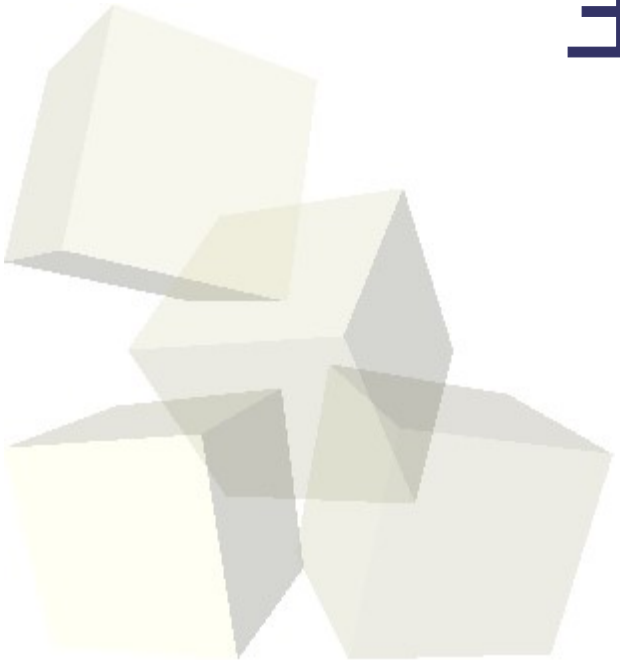
#7 主記憶管理：主記憶管理基礎



復習

- OS
  - 複数プロセスによるリソース使用の調停
  - OS自体とユーザプログラムもリソース共有
- これまで
  - 主にCPUのリソース使用調停について
- これから
  - 主記憶のリソース使用調停について詳しく

7.1
主記憶管理の目的



主記憶の構成

- プログラム間で共有
  - OS
  - ユーザプログラム
  - データ領域
- メモリアドレス
  - 場所を示す番地
  - アドレスは1次元

主記憶の共有

- 一般にマルチプロセス
  - 主記憶は共有資源
  - 複数のプロセスが
    アクセス
- 他のプログラムとの
  調停が必要
  - アクセス時刻だけでなく
  - アクセス領域も

問題点(1):データ保護

- 不正アクセス
  - ユーザプログラムのバグ
    - → 初期化されていない
      ポインタ変数への代入
    - → 配列のサイズを超えた
      要素への代入
  - 他プロセスだけでなく
    OS領域も破壊してしまう
    可能性

問題点(2):有限性

- 主記憶領域は有限
  - そのままではある程度の数のプロセスしか同時処理できない
  - 他にどんなプロセスが動作しているかを意識しながらプログラムを作成するのは無理

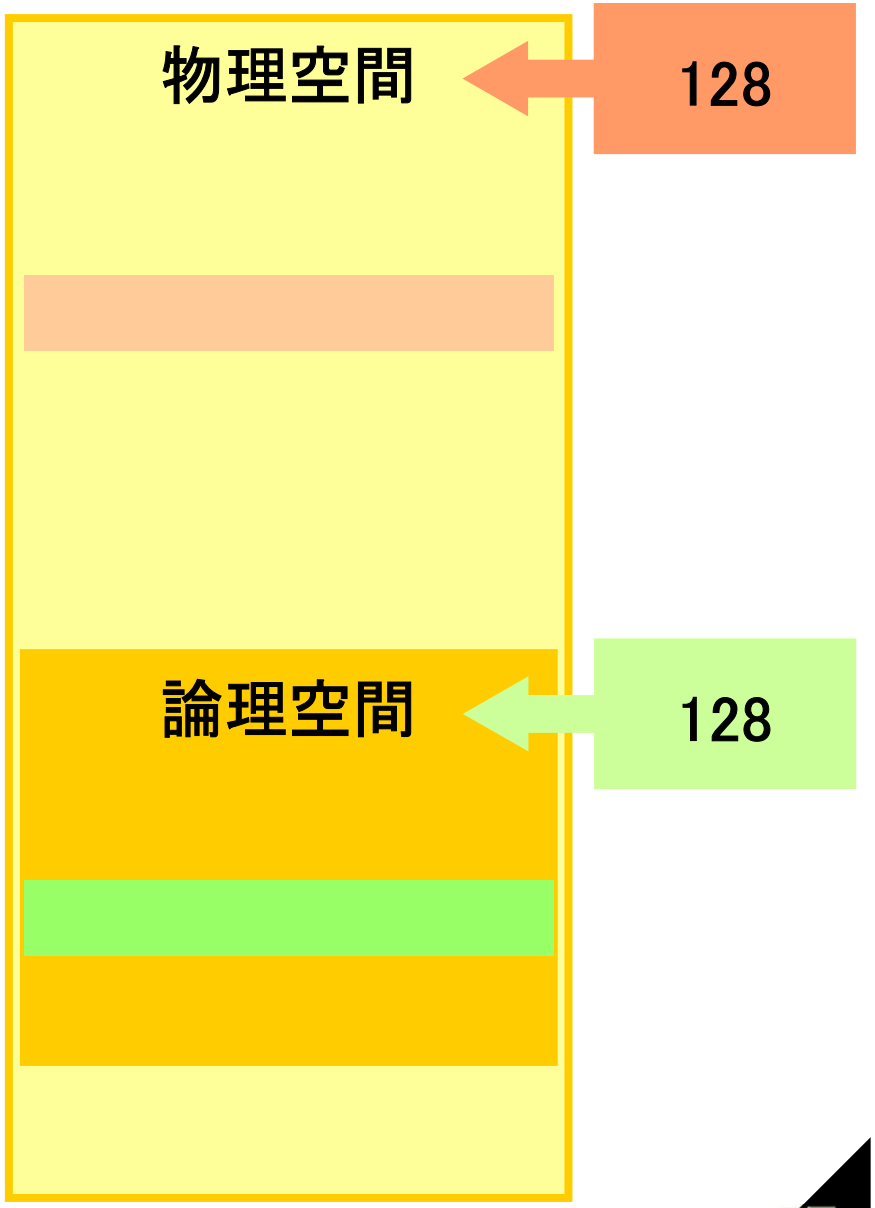
主記憶の仮想化

- **データ保護機能**
  - 他のプロセスからの意図しない読み出し・書き換えを防ぐ必要
  - 自プロセスも，自分のプログラム領域やデータ領域を破壊しないための防御・警告システムが必要
- **主記憶サイズの制限の排除**
  - 主記憶の物理的サイズに関係なく，プロセスに必要なメモリ量を割り当てる必要
  - プログラマに主記憶サイズを意識させない

主記憶も仮想化で対応

アドレス空間

- **物理アドレス空間**
  - 「128番地」は，メモリ上の
    まさに「128番地」を指す
- **論理アドレス空間**
  - 「128番地」は，メモリ上の
    どこかの番地（物理アドレス）を指す
    が，実際の「128番地」
    とは限らない。
    プロセスによって異なる場合も。
  - MMU（Memory Management Unit）
    が，
    論理→物理アドレスの変換を行う



理想的な論理アドレス空間

- 大きさが無制限
  - プロセスは主記憶の空き容量を考慮する必要なし
  - プログラムの簡単化，バグの可能性減少
- プロセスごとに固有
  - 他のプロセスからのアクセスに対し保護
- プログラム部，データ部，スタック部など分離
  - 用途ごとに空間を分けることで，
    自プロセス内での不正アクセスの可能性を低減
- 必要時にはプロセス間で共有も可能
  - 並列動作するプロセス間で共有し，
    高速な通信機構として使用

変数と主記憶

■ 変数

- 数値を記憶するために使用
- 変数に相当する記憶場所が必要 → 主記憶上に確保
- 変数と主記憶上のアドレスとの対応づけが必要

プログラム

```
x = 10;
y = 20;
z = x + y;
:
:
```

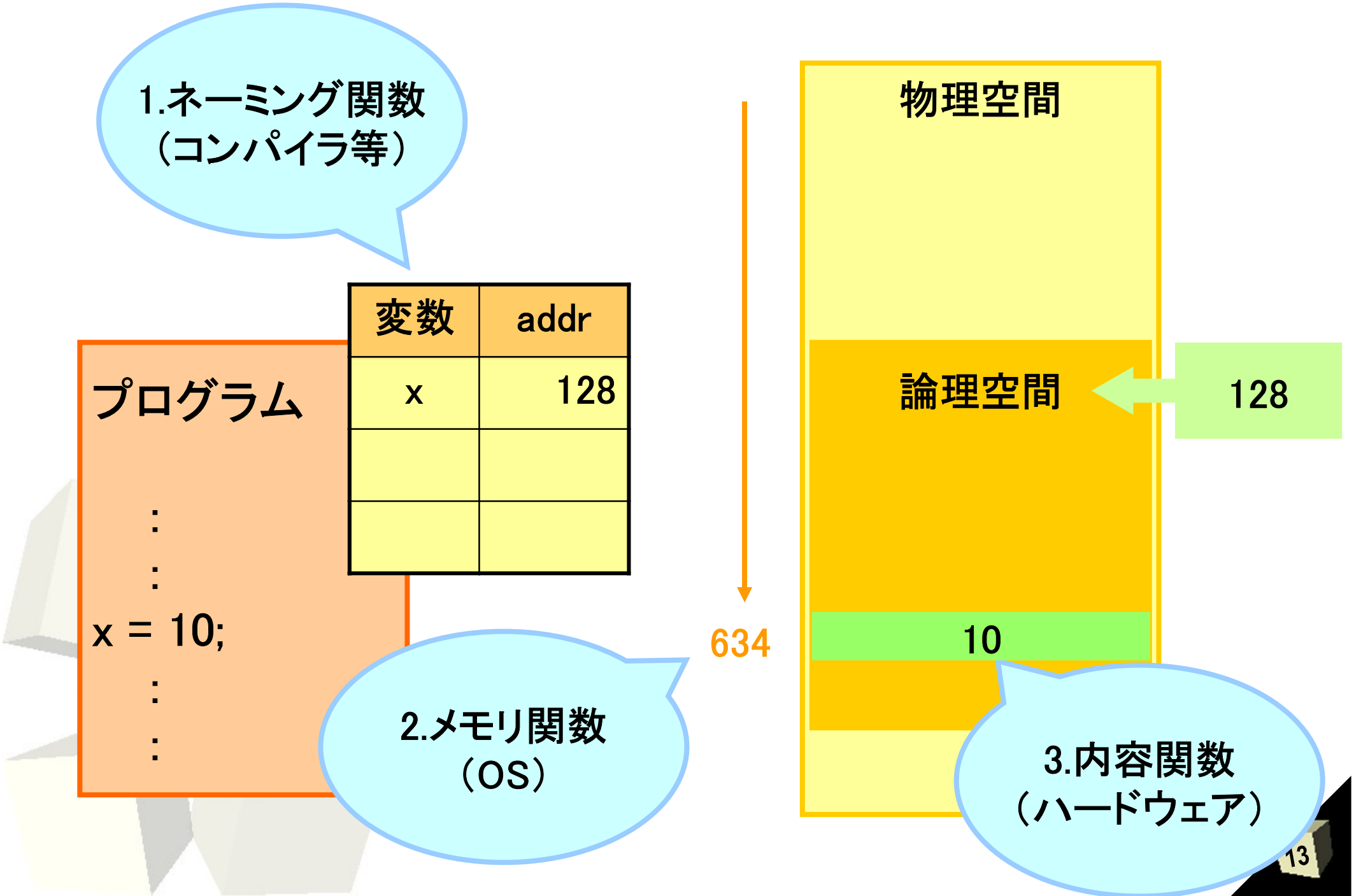
変数	アドレス
x	8923
y	2733
z	4802

コンパイル時

変数アクセス時の様々な変換

- ネーミング関数
  - 変数，定数などの識別子を論理アドレスに変換
  - コンパイル・リンク時に行われる
- メモリ関数
  - 論理アドレスから物理アドレスに変換
  - OSによって行われる
- 内容関数
  - 物理アドレスから，そのアドレスに格納された内容に変換
  - ハードウェアによって行われる

変数アクセス時の様々な変換



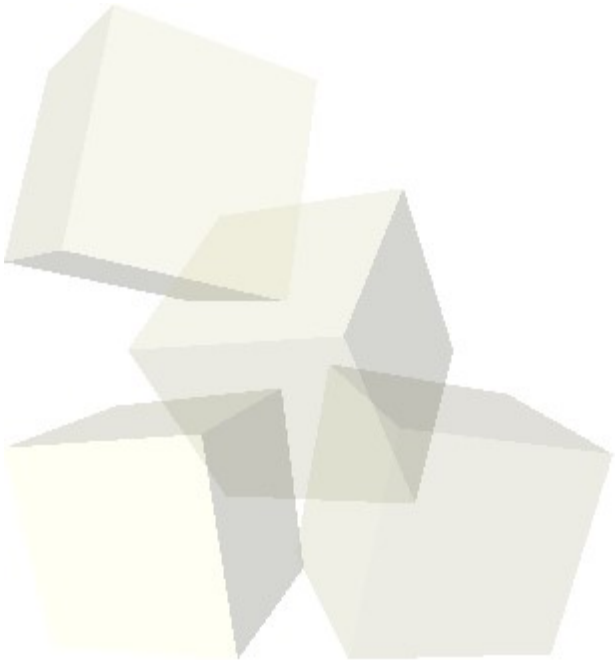
変数アクセス時の様々な変換

- ネーミング関数
  - 変数，定数などの識別子を論理アドレスに変換
  - コンパイル・リンク時に行われる
- メモリ関数
  - 論理アドレスから物理アドレスに変換
  - OSによって行われる
- 内容関数
  - 物理アドレスから，そのアドレスに格納された内容に変換
  - ハードウェアによって行われる

メモリ関数の設計

- メモリ関数の処理
  - 遅いと，プログラムの実行速度に大きく影響
- メモリ関数に要求されること
  - 論理アドレスから物理アドレスへの変換を高速に
  - ハードウェアを簡潔に
  - プログラマに使いやすい論理アドレス空間を提供

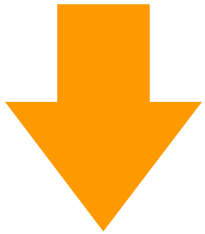
7.2
下限レジスタ機構



目的

- 主記憶空間の他プロセスからの保護
  - プログラム領域を書き換えられると，
    プロセスの誤動作・停止などの危険

- OS領域
  - 他プロセスからの保護が特に重要
  - システム全体の停止につながる



  - ユーザプログラムの使用領域と
    OSの使用領域を厳密に分ける必要

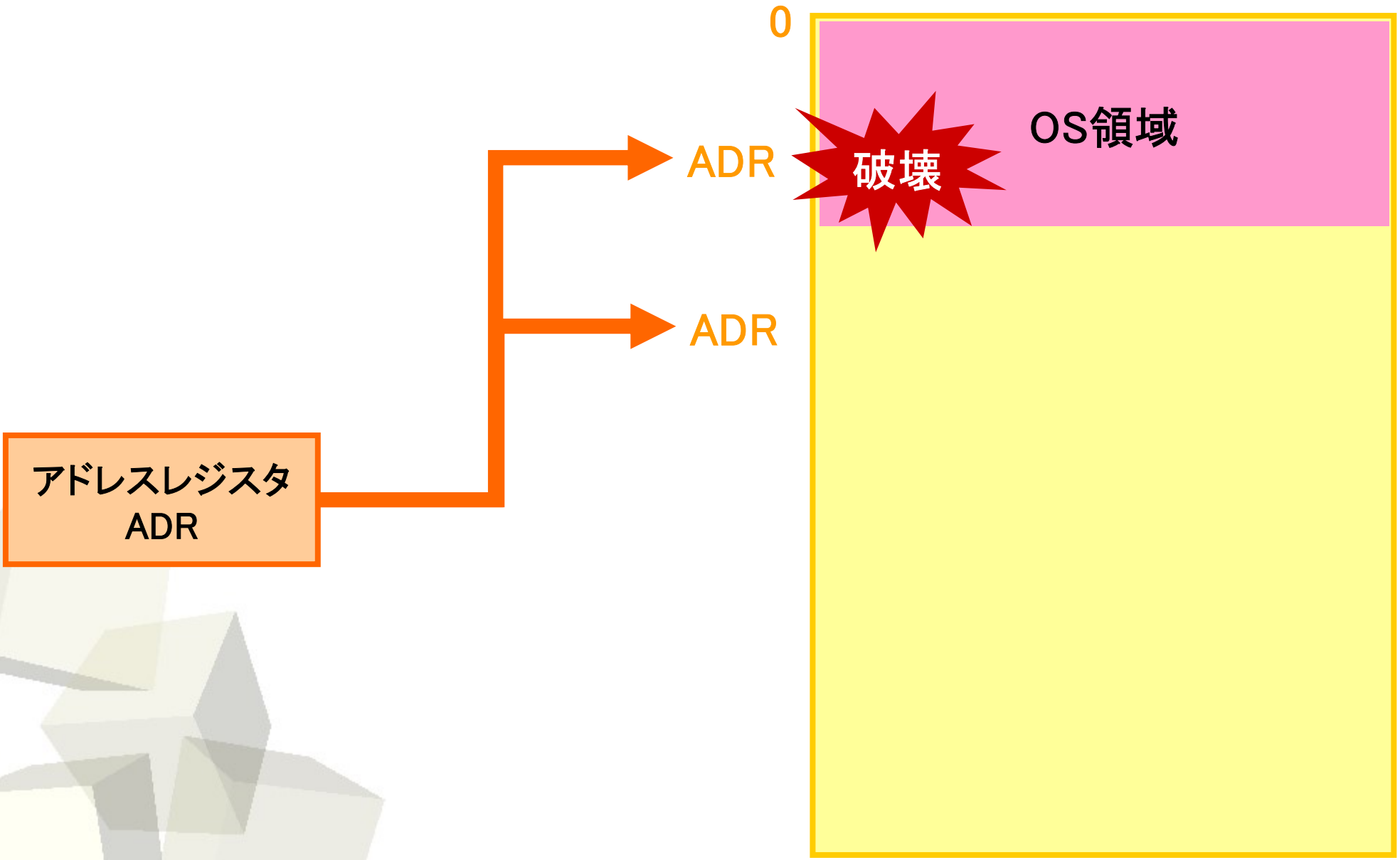
下限レジスタ機構の仕組み

- 方針
  - ユーザプログラムが使用できる記憶領域の下限を設定
  - ユーザプログラムによるアクセスかOSによるアクセスかは，実行モードで判断

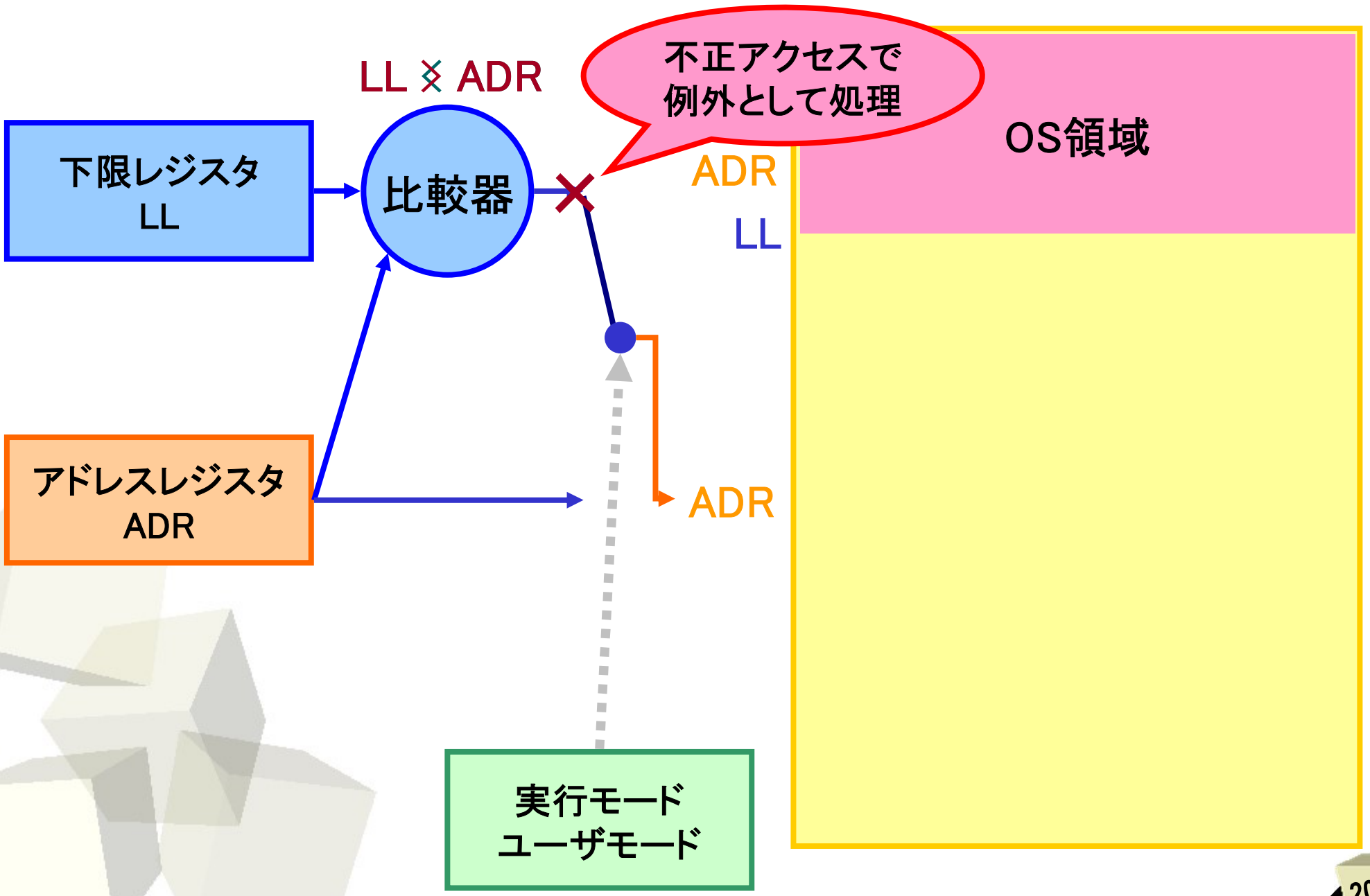
復　習

スーパバイザモード
　OSを実行するモード
　CPU内の全てのリソースを利用可能
ユーザモード:
　アプリケーションを実行するモード
　利用できるリソースに制限あり

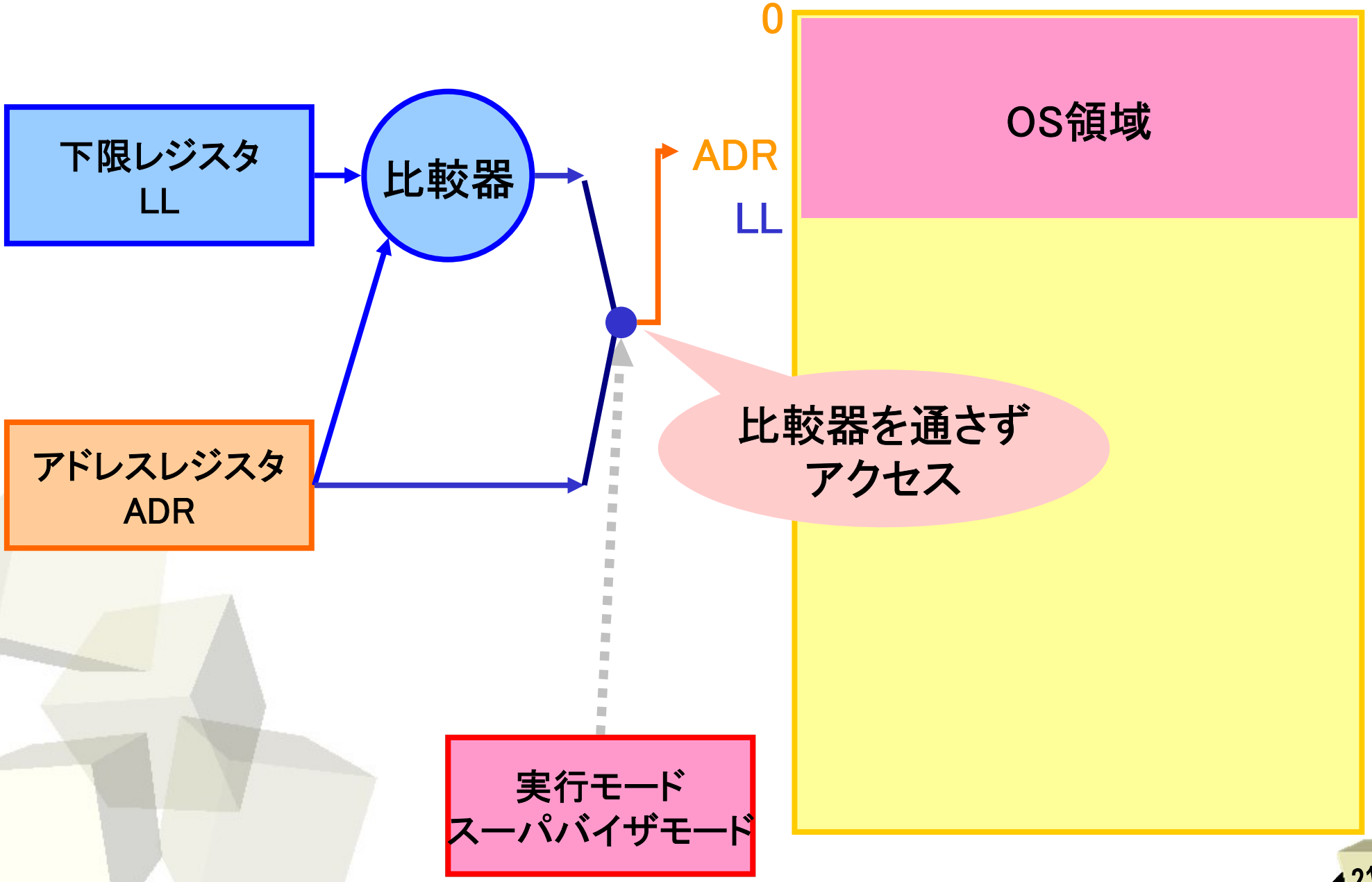
下限レジスタ機構の仕組み



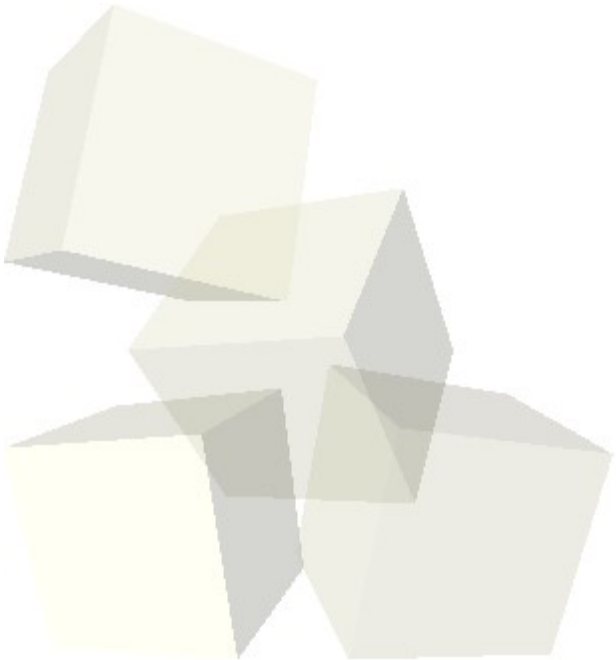
下限レジスタ機構の仕組み



下限レジスタ機構の仕組み



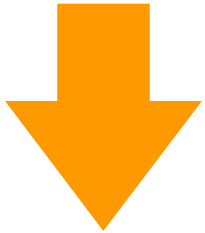
7.3
ロック／キー機構



下限レジスタ機構とその問題

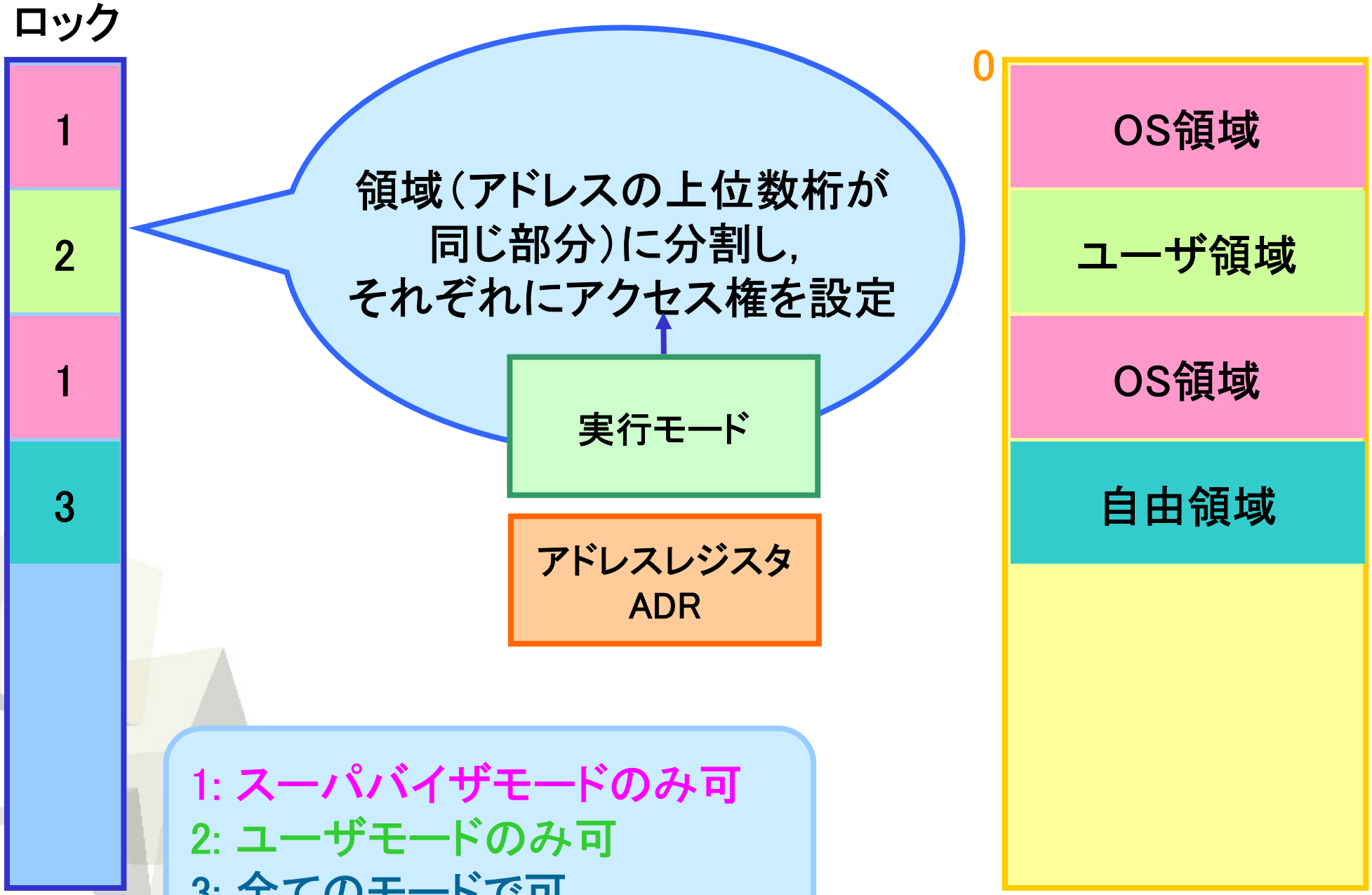
- ユーザ領域の下限を設定
  - 下限レジスタが示す境界で OS/ユーザ領域を区別

- 問題点
  - 領域境界が1つしかない
  - OS領域を保護することしかできない
  - 複数のプロセス間でアクセス権は設定できない

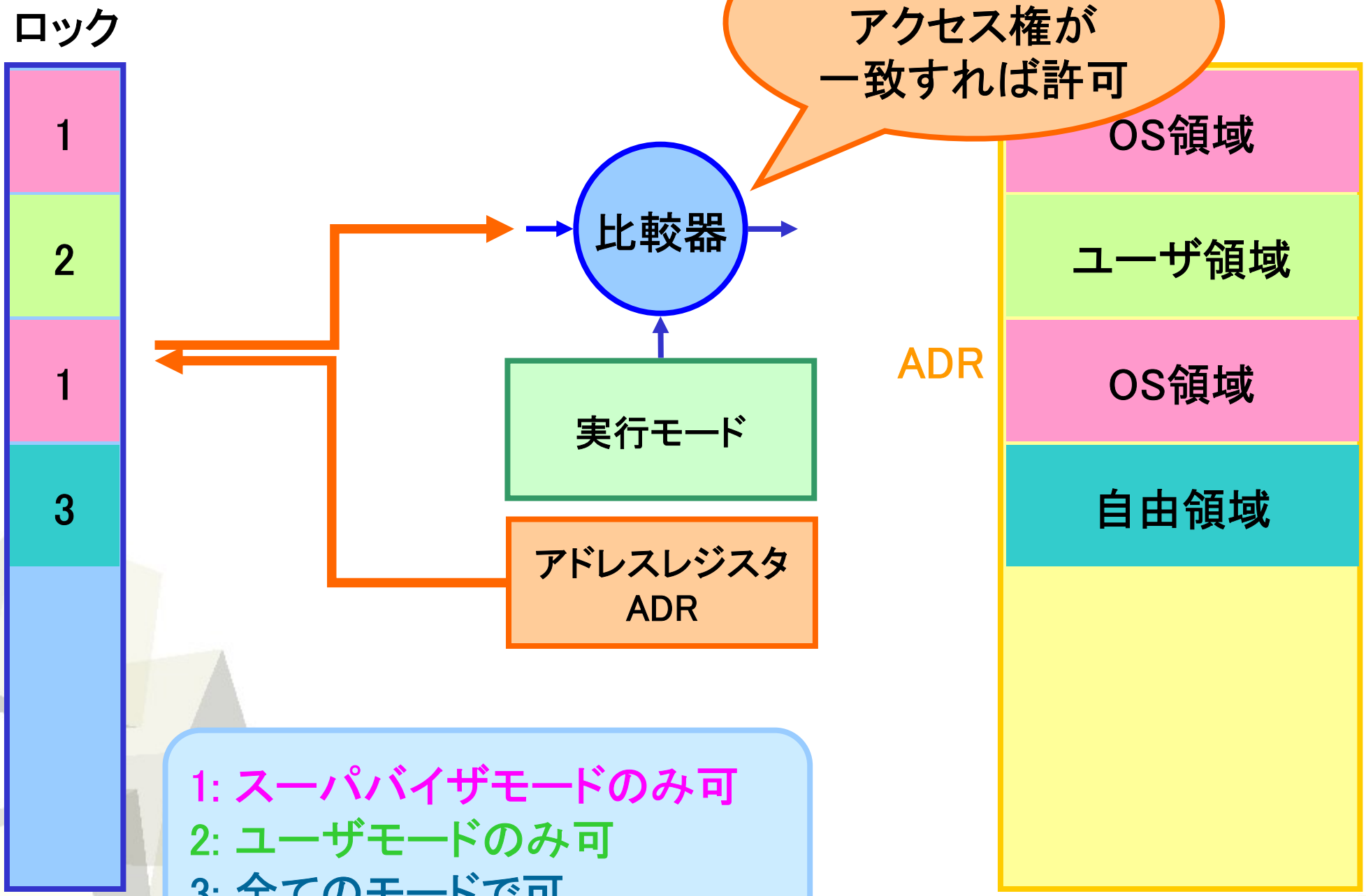


  - 任意・複数の境界を設定し，プロセスごとに
    アクセス権を設定したい

ロック／キー機構の仕組み



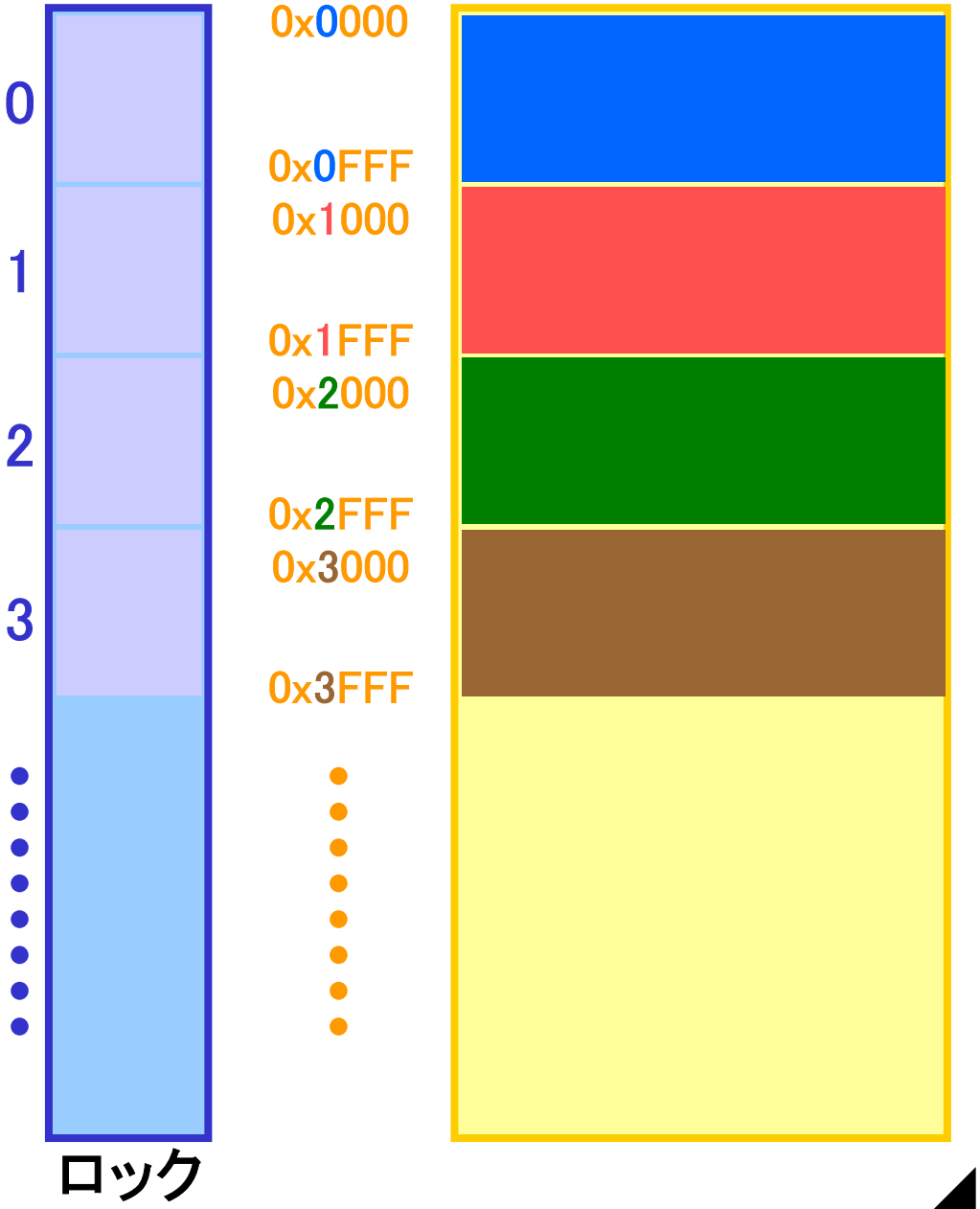
ロック／キー機構の仕組み



ロック機構における領域分割

- アドレスを分割
  - アドレス上位数桁
  - アドレス上位が同じ領域を同じ属性を持つ領域とする
  - このアドレス上位部を

  ロック表のインデックスとして使用



まとめ

- 主記憶管理
  - ユーザに独立した論理アドレス空間を提供
  - 論理アドレス空間の要件
    - → 無限大
    - → プロセスごとに固有
    - → プロセス間で共有可能
    - → 複数の1次元アドレス（プログラム部，データ部等の分離）

まとめ

- 下限レジスタ機構
  - ユーザ領域の下限を設定
  - OS領域とユーザ領域を分離
  - 実行モードに応じてアクセス可否を判断

- ロック／キー機構
  - アドレスの上位数桁でメモリ領域を分割
  - 各領域ごとにアクセス権限を設定
  - アドレスに応じた権限（ロック値）と実行モードからアクセス可否を判断