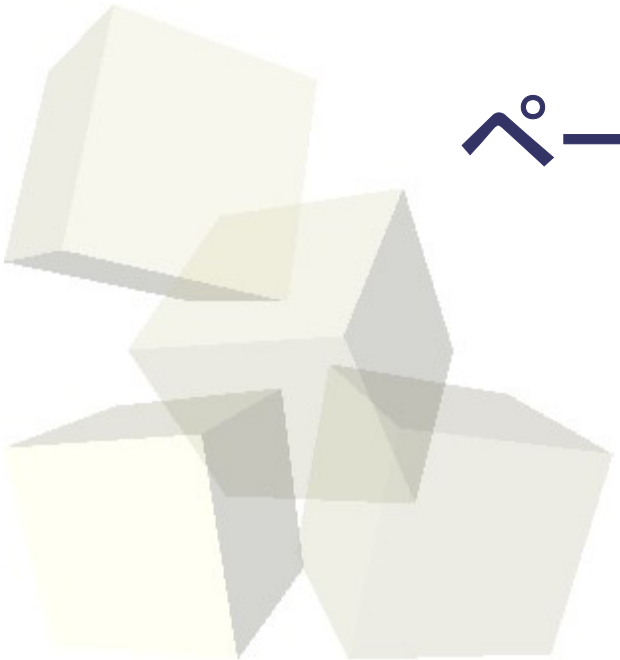
オペレーティングシステム

#10 主記憶管理：
セグメンテーション，
ページ化セグメンテーション



復習

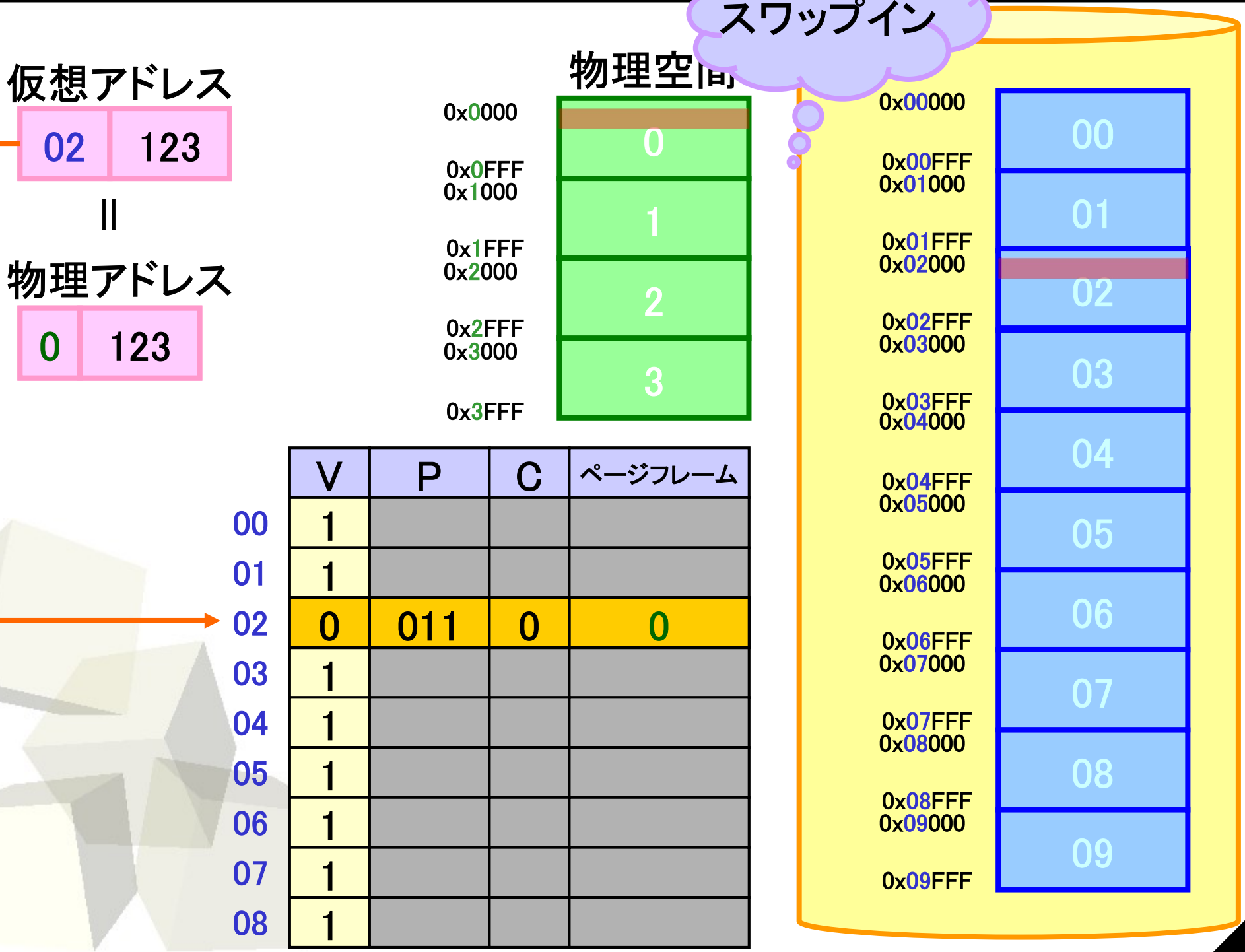
ページング

- 主記憶の再配置システム
- 仮想アドレスの上位(ページ番号)を
  物理アドレスの上位(ページフレーム番号)に
  変換することでアドレス変換
- フラグメンテーションも改善

ページテーブル

- 上記変換を行うためのテーブル
- ページに対応するエントリごとにフラグなどを配置
  - → ページへのアクセス権などを設定可能

復習：ページングの動作



	V	P	C	ページフレーム
00	1			
01	1			
02	0	011	0	0
03	1			
04	1			
05	1			
06	1			
07	1			
08	1			

復習

ページングの改善法

- ハッシュ関数
  - → ページテーブルの巨大さを緩和
  - → 仮想アドレス空間に比例する大きさが必要だったのが，物理アドレス空間に比例する大きさまで縮小
- 連想レジスタ
  - → 主記憶へのアクセス回数の緩和
  - → 最近使用した「ページ番号→ページフレーム番号」の変換結果をCPU内で記憶
  - → ページテーブル検索のための主記憶アクセスを削減

今日の内容

- ページングの構成法
  - 多重レベルページング (10.3)
  - 0レベルページング (10.4)
- セグメンテーション
  - セグメンテーション (10.1)
  - ページ化セグメンテーション (10.2)
- 教科書とは違う順番でやります

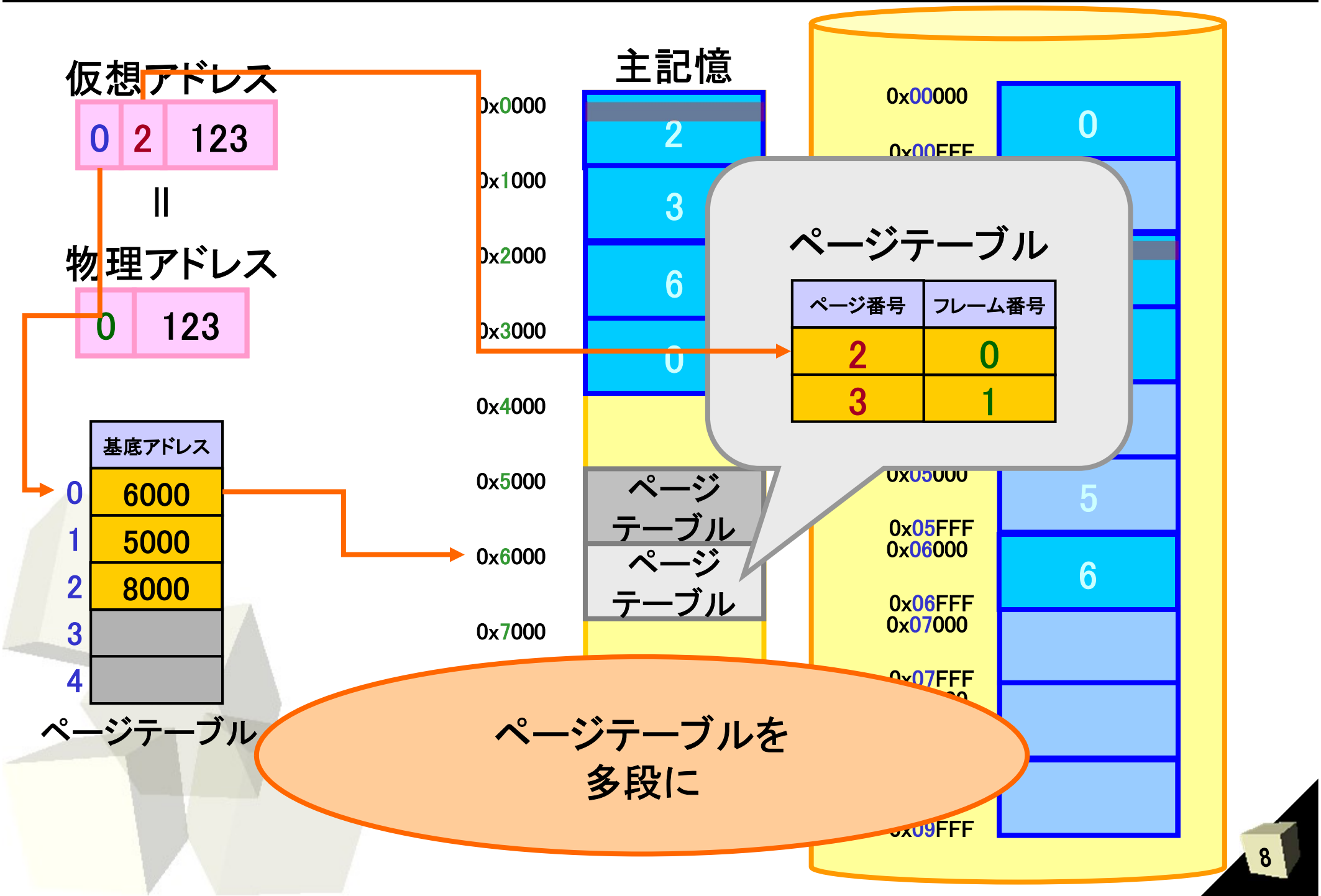
10.3
多重レベルページング

復習：ページングの問題点

- ページテーブルが巨大
  - （仮想記憶の大きさと同じエントリ数）
    ×（同時実行プロセス数）
    だけのエントリ数が必要
  - 主記憶を圧迫

- 前回はハッシュ関数を利用することで改善
  - → 仮想アドレス空間に比例する大きさが必要だったのが，
    物理アドレス空間に比例する大きさまで縮小
  - さらに改善したい

多重レベルページング


仮想アドレス
0
2
123
=
物理アドレス
0
123
基底アドレス
0
6000
1
5000
2
8000
3
4
ページテーブル
主記憶
0x0000
0x1000
0x2000
0x3000
0x4000
0x5000
0x6000
0x7000
2
3
6
0
ページ
テーブル
ページ
テーブル
ページテーブル
ページ番号
フレーム番号
2
0
3
1
0x00000
0x00FFF
0
0x05FFF
0x06000
0x06FFF
0x07000
0x07FFF
0x09FFF
5
6
ページテーブルを
多段に

多重レベルページング

仮想アドレス

0	2	123

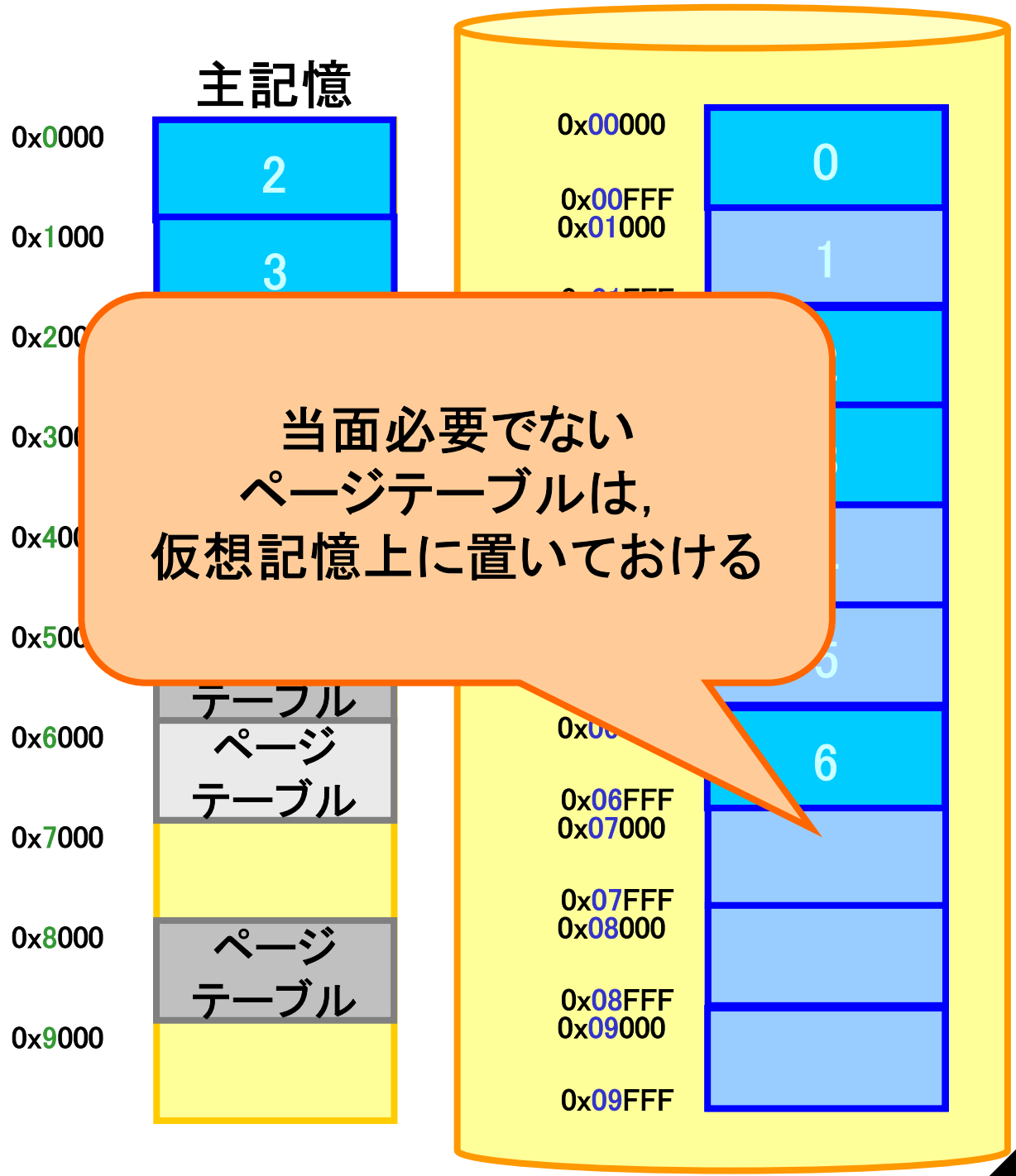
=

物理アドレス

0	123

	基底アドレス
0	6000
1	5000
2	8000
3	
4	

ページテーブル

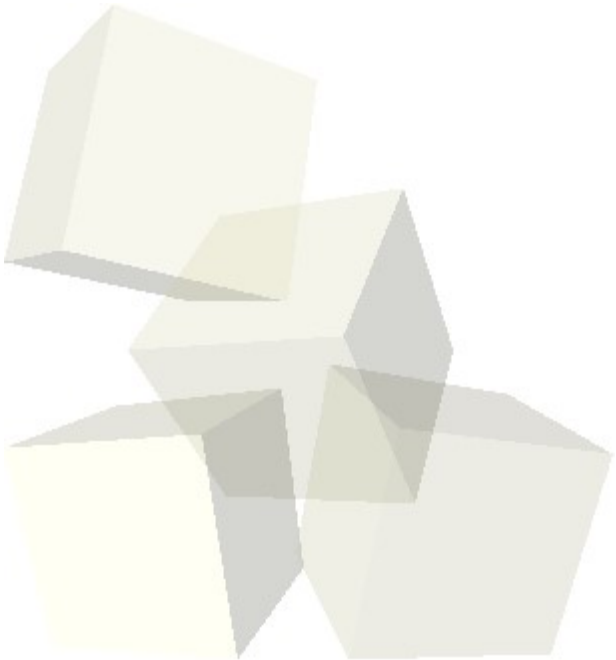


当面必要でない
ページテーブルは，
仮想記憶上に置いておける

多重レベルページング

- ページングを多段化
  - 2段だけでなく，さらに多段化も可能
- ページテーブルを細かく分割可能
  - 必要なテーブルのみを主記憶上に置くことで主記憶使用量を削減
- 多段化により，主記憶アクセスは増加
  - TLBにより解決

10.4
0レベルページング
（連想写像方式）

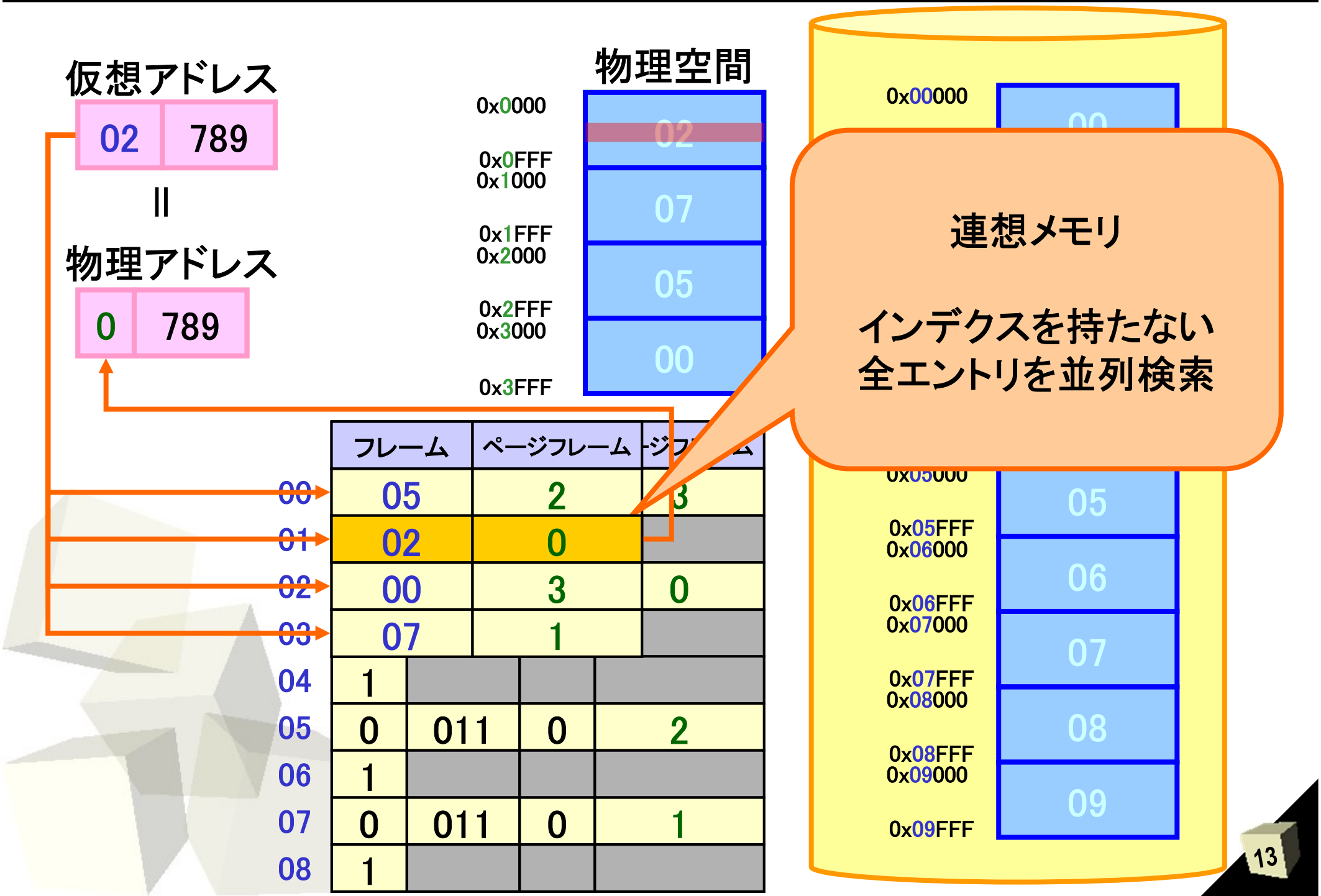


連想写像方式

- 一般的なページング
  - 主記憶上にページテーブル
  - CPU内のMMUが，ページテーブルを用いて
    アドレス変換

- 0レベルページング（連想写像方式）
  - ハードウェアとしてのMMUを持たない
  - 一般的なページテーブルを持たず，
    連想メモリで構成したTLBでアドレス変換

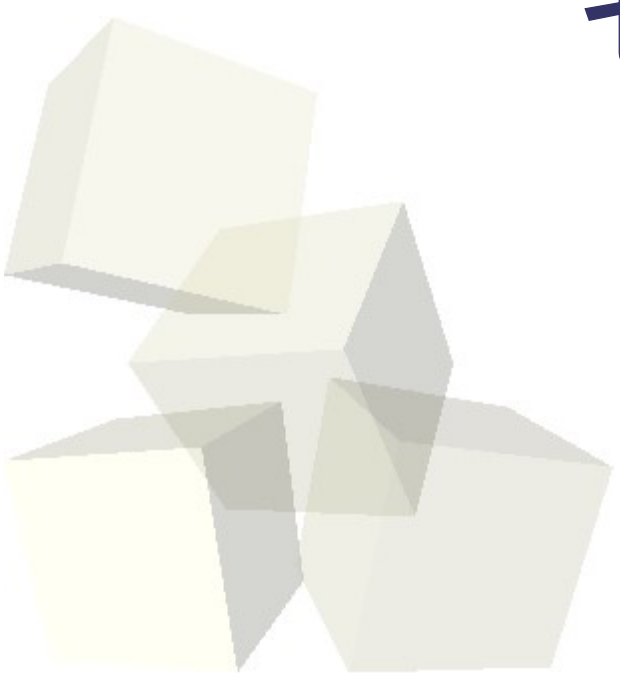
0レベルページングの動作



0レベルページングの特徴

- メリット
  - 高クロック実装が可能
    - → MMU不要のため，CPU機構が単純化
- デメリット
  - TLBヒットしなかった場合のオーバヘッドが膨大
    - → ソフトウェア処理のため
- よって…
  - 主記憶使用量の少ないプログラムには高速
  - 主記憶使用量の多いプログラムには非常に低速

10.1
セグメンテーション



復習：理想的な論理アドレス空間

- 大きさが無制限　　ページングで実現可能
  - 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

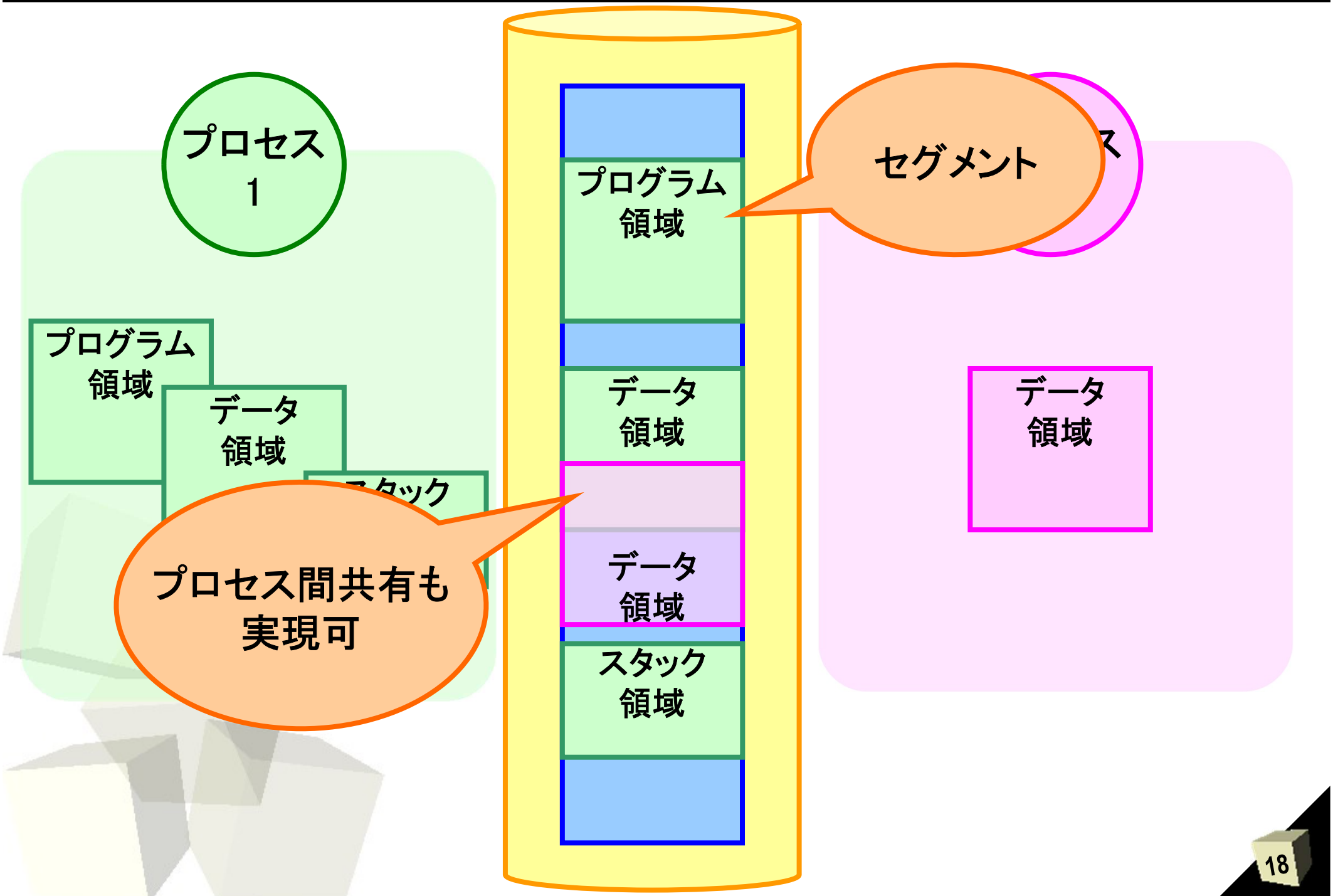
こちらも実現したい

セグメンテーション

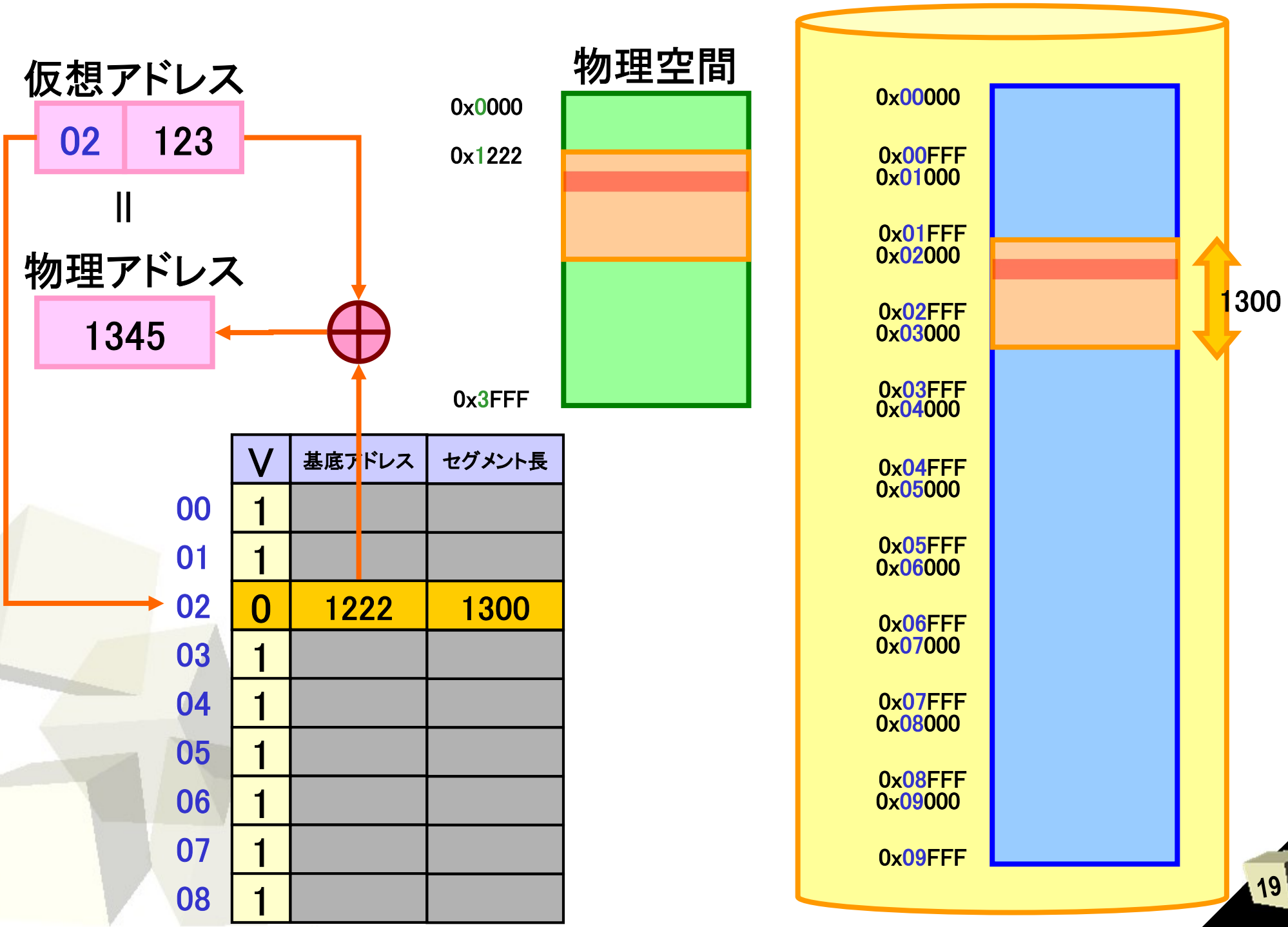
- ページ（ページング）
  - 一定の大きさを割り当て単位とする

- セグメント（セグメンテーション）
  - プロセスに対し複数のセグメントを割り当て
    - → 各セグメントは論理的に独立
    - → プログラム部，データ部など固有領域として使用可能
  - 各セグメントはその論理空間の大きさを自由に増減可能

セグメンテーション


プロセス
1
プログラム
領域
データ
領域
プロセス間共有も
実現可
セグメント
プログラム
領域
データ
領域
データ
領域
スタック
領域
データ
領域

セグメンテーション

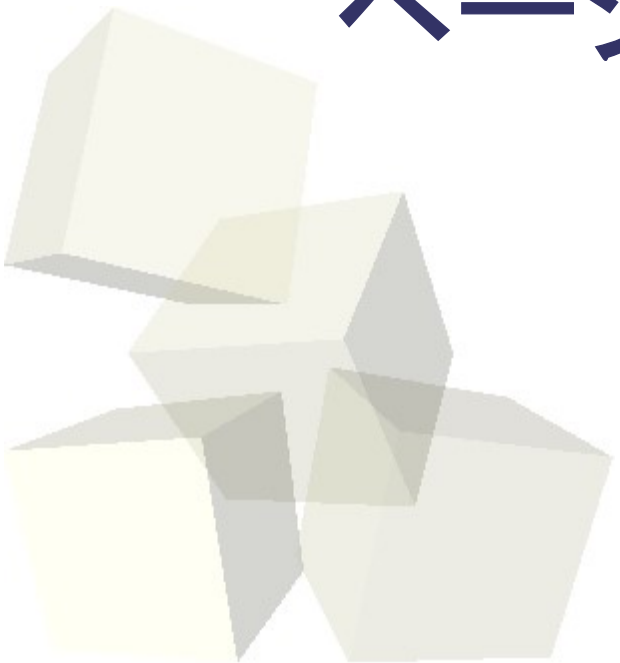

仮想アドレス
02
123
=
物理アドレス
1345
物理空間
0x0000
0x1222
0x3FFF
V
基底アドレス
セグメント長
00 1
01 1
02 0 1222 1300
03 1
04 1
05 1
06 1
07 1
08 1
0x00000
0x00FFF
0x01000
0x01FFF
0x02000
0x02FFF
0x03000
0x03FFF
0x04000
0x04FFF
0x05000
0x05FFF
0x06000
0x06FFF
0x07000
0x07FFF
0x08000
0x08FFF
0x09000
0x09FFF
1300

セグメンテーション

- 利点
  - プログラム部，データ部など，用途別に複数をプロセスに割り当て
  - 各セグメンテーションは大きさを増減可
- 欠点
  - フラグメンテーション

他の手法と組み合わせて利用

10.2
ページ化セグメンテーション



ページング v.s. セグメンテーション

ページング

- 固定長ブロック
  - → フラグメンテーションの解決
- ページテーブル
  - → 仮想→物理アドレスの1対1対応
  - → 動的再配置による仮想記憶の実現

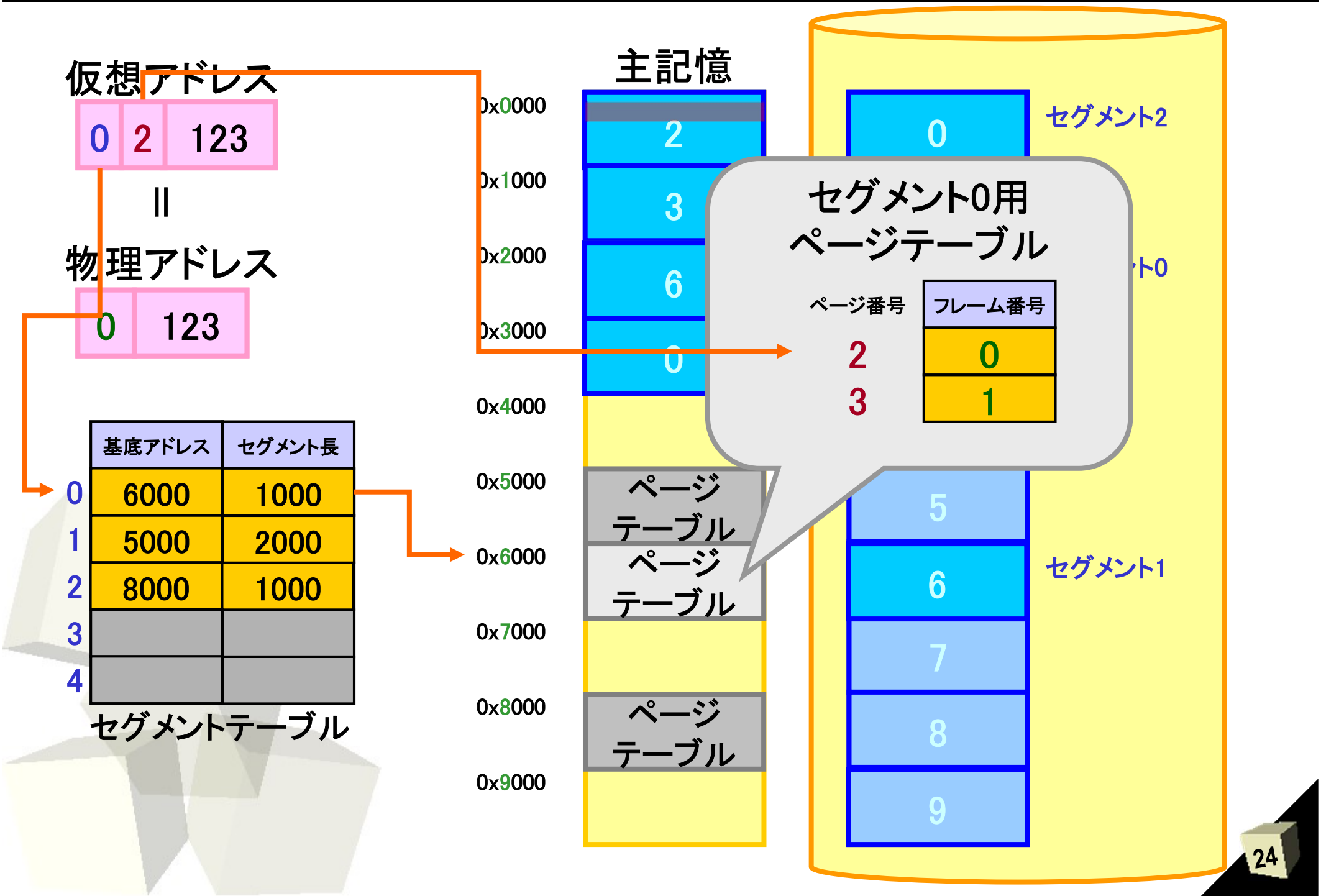
セグメンテーション

- 複数のアドレス空間
  - → 用途別論理空間の実現
- アドレス空間のオーバラップ
  - → 複数プロセスによる空間の共有を実現

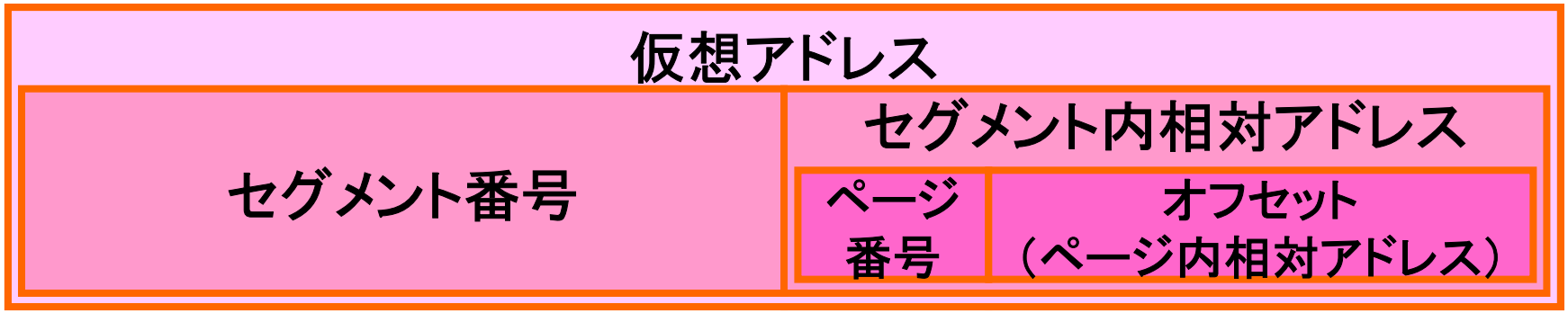
ページ化セグメンテーション

- ページング ＋ セグメンテーション
  - セグメンテーションを，
    複数のページにより構成する
  - セグメントごとにページテーブルを用意

ページ化セグメンテーション



■ 仮想アドレスの構造



- セグメントがページ単位で構成されている
  - → セグメンテーションとページングの階層化
- セグメント内相対アドレスが，
  ページングにおける仮想アドレスに相当
  - → 仮想アドレスの構造も二階層になっていると考えるとわかりやすい

ページ化セグメンテーションの利点

- フラグメンテーションの回避
  - 主記憶割り当ては基本的にページ単位
- 複数セグメント
  - 各セグメントは大きさ増減可能
  - 複数使用により，用途別に使い分け可能
- プロセス間共有
  - セグメンテーションとほぼ同様に共有可能
- ページテーブルの分散
  - ページテーブルが複数に分割されるので，
    多重レベルページング同様，その一部を
    仮想記憶に追い出すことで主記憶使用量削減

今日のまとめ

- 多重レベルページング
  - 従来のページングにおける，仮想アドレスの「ページ番号部」を複数に分割
  - それぞれをページ番号とし階層化したテーブルを検索
  - ページテーブルの分割により，テーブルによる主記憶使用量を削減（残りは仮想記憶へ）
- 0レベルページング（連想写像方式）
  - MMUを持たず，CPUの構成を単純化（＝高速化）
  - 連想記憶（検索のためのインデクスを持たない）でTLBを構成

今日のまとめ

- セグメンテーション
  - セグメント:
    仮想記憶と物理記憶を対応させる可変長な単位
  - プロセスあたり複数のセグメントを許す
  - セグメント間のオーバラップ(共有領域)を許す
- ページ化セグメンテーション
  - ページングとセグメンテーションの利点を融合
  - セグメンテーションの利点を生かしつつ
    - → フラグメンテーションを回避
    - → ページテーブルの分割により，多重ページング同様
      主記憶使用量の削減