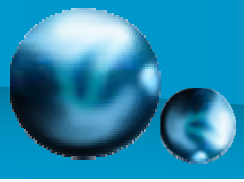
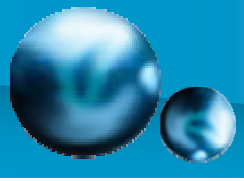
オペレーティングシステム
3章 メモリ管理
UNIXにおける実装

大阪大学大学院情報科学研究科
村田正幸
murata@ist.osaka-u.ac.jp
http://www.anarg.jp/



UNIXの実装
メモリ管理の実装

- 初期のUNIX：スワッピング方式
  - メモリ不足のとき
    - いくつかのプロセスのメモリイメージのすべてをスワップアウト
  - メモリに余裕ができたとき
    - プロセスイメージ全体をスワップイン
- 近代のUNIX(4.2BSD～)
  - 仮想記憶（デマンドページング；要求時フェッチ）とスワッピングの併用

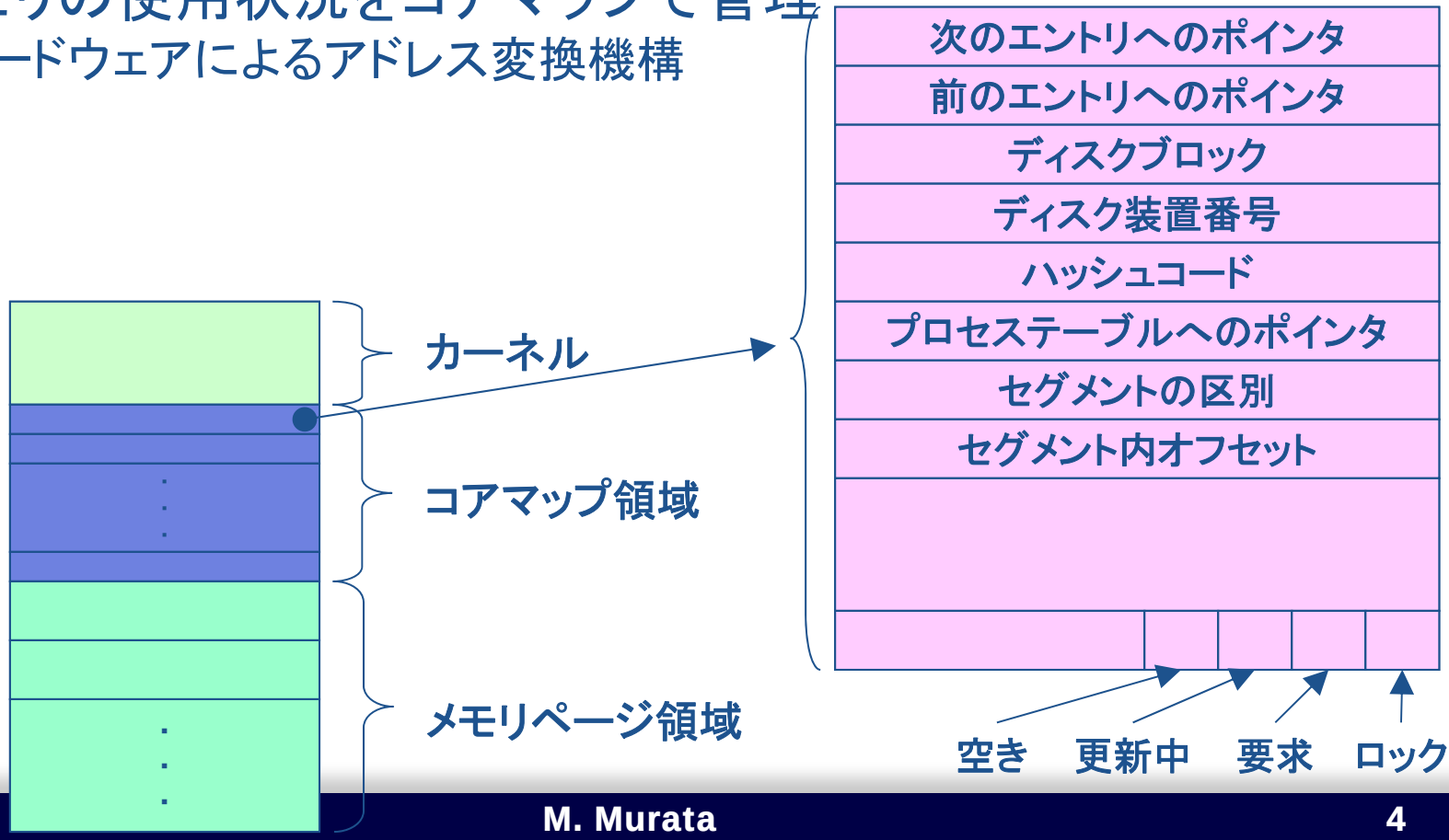


UNIXにおけるスワッピング

- スワッピングはシステムプロセスのスワッパ(PID=0)が担当
  - データセグメントを持たないシステムプロセス
    - 最大の優先度でsleepしている
  - 起動
    - プロセス生成時
    - メモリ容量が足りなくなった時
- スワップアウト対象プロセス
  - ディスクI/Oなどの遅いイベントを待っているプロセスの中で
    - 優先度が低い
    - メモリに長く滞在している
    - サイズの大きいもの
  - なければ、実行可能状態のプロセスの中から、同様の基準で選ぶ
  - スラッシングを軽減するため、ページングですべてページアウトされたプロセスも対象とする
- スワップイン対象プロセス
  - 優先度が高く、長くスワップアウトされていたもの
    - サイズの小さいもの
  - メモリが確保されるまで、スワップアウトを繰り返す

デマンドページング

- カーネルとコアマップ以外を対象にするページング方式
  - ページ枠の大きさは512B～4KB
- 実メモリの使用状況をコアマップで管理
  - ハードウェアによるアドレス変換機構



動的アドレス変換機構の例

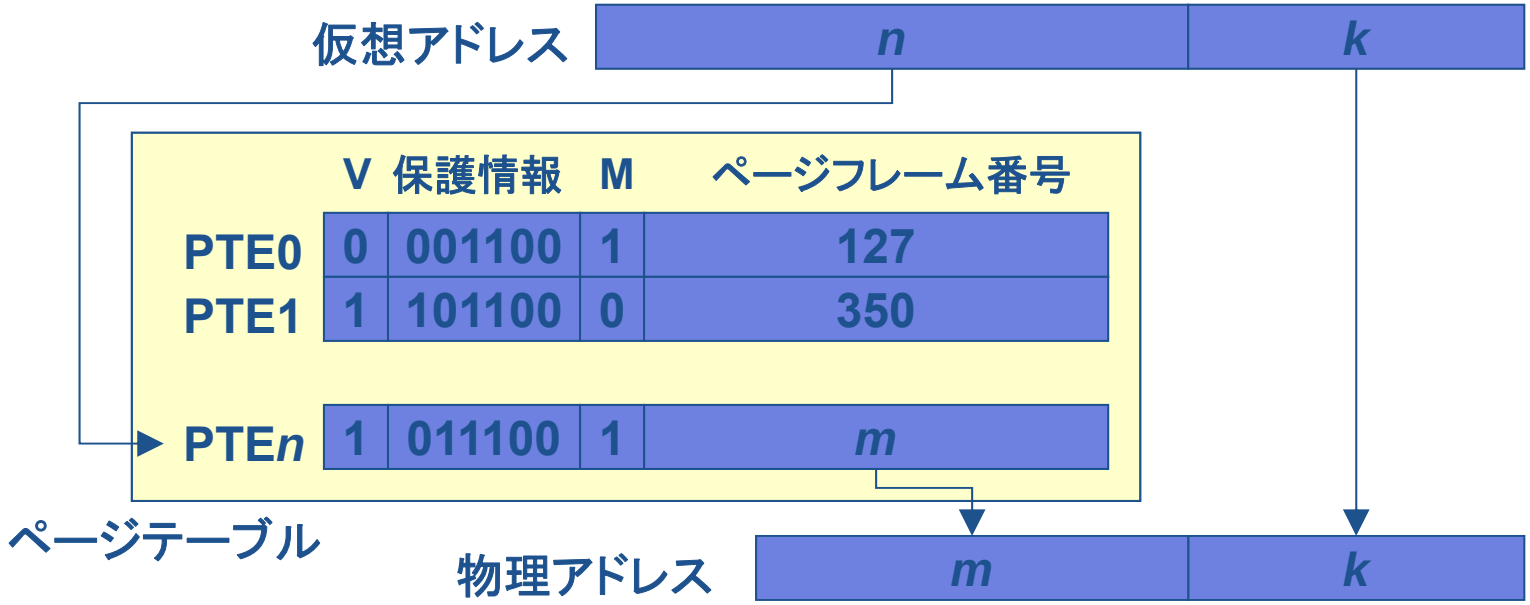
- VAX-11の例
- テキスト(＋データ)セグメント、スタックセグメント、カーネルをそれぞれ別の仮想空間におく(ページ化セグメント方式)
  - 独立したページテーブル(PTE)で管理、ページサイズは512B
- ユーザプロセスのページテーブル
  - テキスト+データセグメント用のページテーブルはカーネル仮想空間におかれ、ページアウトの対象とする(当時はメモリが高価だったため)
  - テキストセグメント用ページテーブルは、カーネルの仮想空間に存在し、ページングの対象になる
    - ページテーブルは大きくてもよい
    - 参照するためには、カーネル空間用のページテーブル→テキストセグメント用のページテーブルの2回のメモリアクセスが必要になる

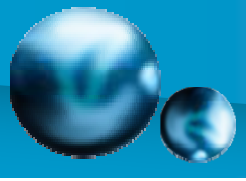
    ⇒PTEキャッシュを行う

アドレス変換機構

- 仮想アドレス（上位 *n*）へのアクセスがあった時、以下を実行する

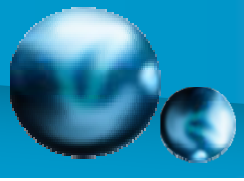
1. テキストセグメント用ページテーブルの該当エントリを参照する
2. PTEのVビット（ページが有効か）を参照
   - 0の時：実メモリに割り付けられていない
     - ページフォールト割り込みによって、ページ内容を読み出す
3. 保護ビットを参照して、アクセス可否判断
   - 許可されていれば、ページフレーム番号 *m* に置き換える





ページデーモン

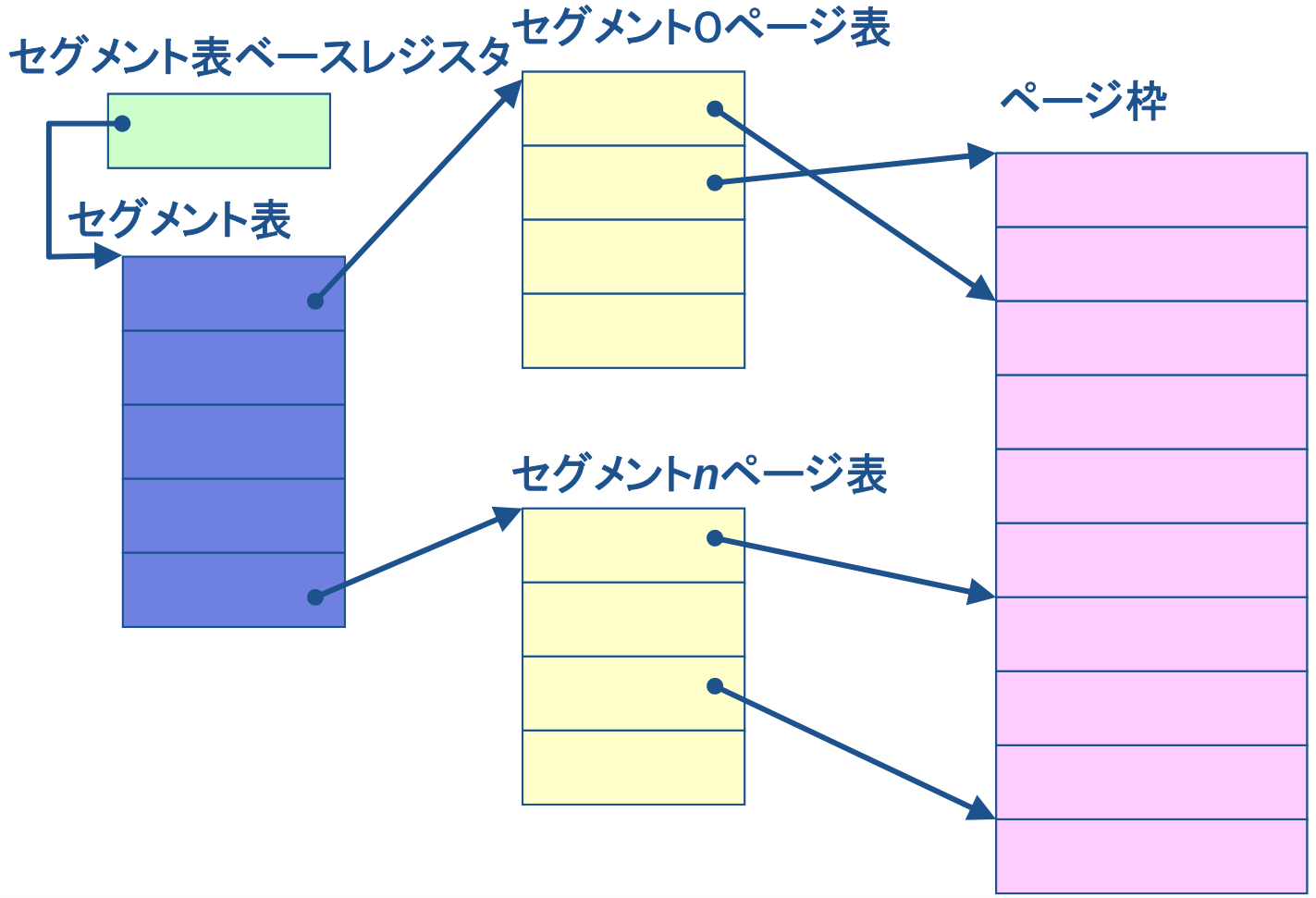
- 空きページテーブルを作るシステムプロセス
  - 空きページ枠がないときに起動される、また250msごとに調査
  - 空きページ枠数が所定の閾値以下ならばページアウトを行う（例えばメモリ量の1/4）
  - 「所定の閾値」？
    - 大きすぎるとページが空きになり無駄
    - 小さすぎると待ち時間が大きくなる
    - min(メモリ量/8、256KB)
    - min(メモリ量/8、200KB + MAXUSERS × 10KB)



仮想記憶の復習

- ページング方式とセグメンテーション方式
  - アドレス変換
    - ディレクトリ方式とページ表方式
  - ページ表の多段構成
- 今の主流は、ページセグメンテーション方式
  - コード、データ、スタックごとにセグメントとし、それぞれのセグメントを別の仮想アドレス空間に置き、ページ化する
  - 多段ページ表の一段目をセグメントと考える
  - セグメントが独立した仮想空間を有する多重仮想記憶方式
  - 仮想アドレス＝(セグメント番号、ページ番号、オフセット)
- ページテーブル参照の高速化
  - セグメントテーブル参照、ページテーブル参照へのメモリアクセスが必要になる
  - ページテーブル(PTE)のキャッシュ
  - TLB (Translation Look-aside Buffer)

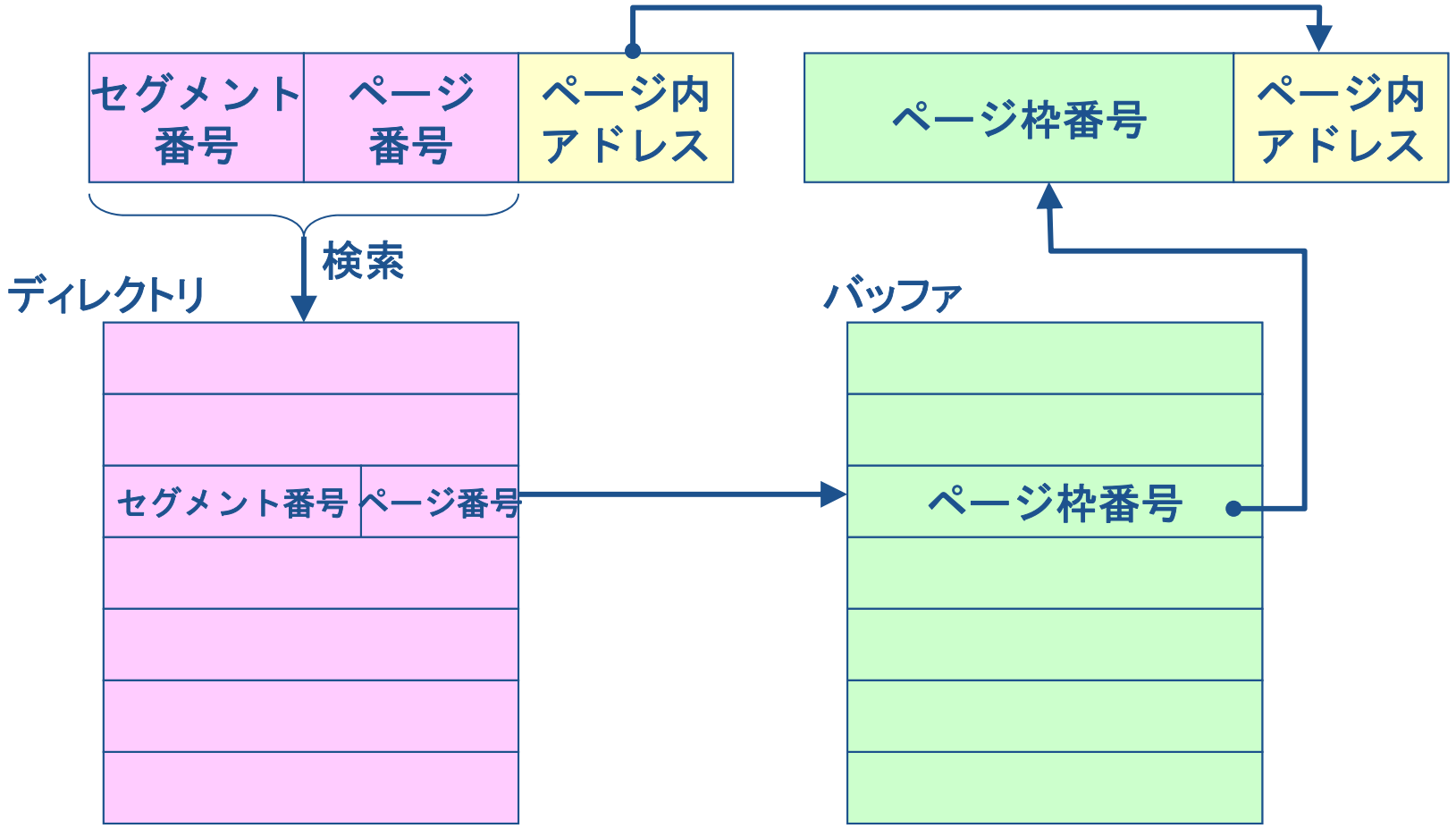
ページセグメンテーション方式



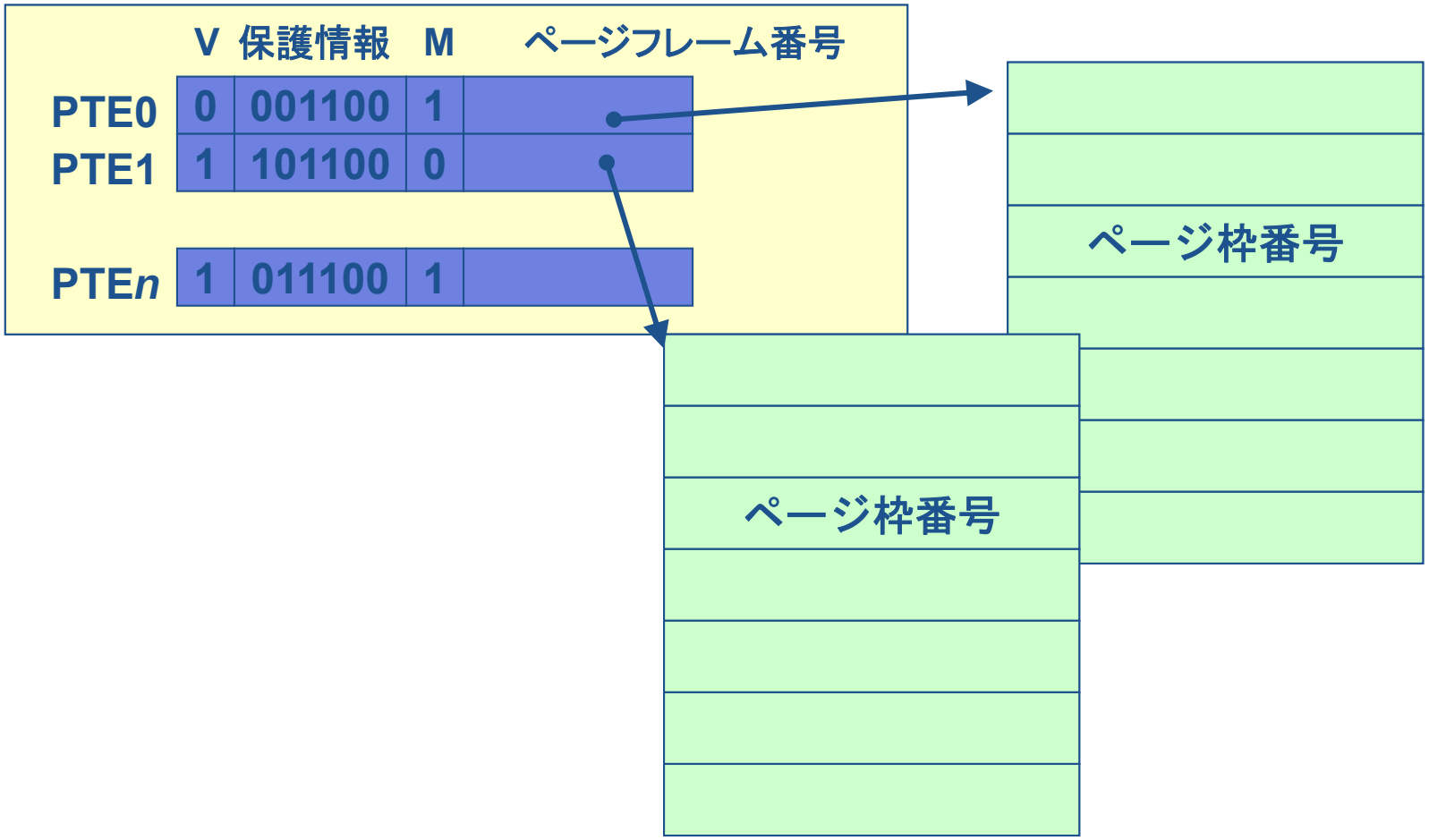
TLB機構

- 連想記憶によって、アドレス変換の高速化を図る
  - 最近行ったアドレス変換の結果を(セグメント番号、ページ番号)とページ枠番号の対で記憶し、引き続くアドレス変換をTLBのページ番号による検索に置き換える
  - 検索した結果、一致したページ番号があればページ枠番号を読み出し、ページ内アドレスを加算して実アドレスを得る
  - 一致するページ番号がなければページ表にアクセスしてアドレス変換を行い、変換結果をTLBに追加する
  - 空きエントリがなければ、エントリの置き換えを行う
- キャッシュメモリとの類似に注意、ただし、ページ単位の場合、少量でも「キャッシュミス」は起こりにくい
- 連想記憶の段数削減のために群連想方式を用いる

TLBの構成


セグメント番号
ページ番号
ページ内アドレス
ページ枠番号
ページ内アドレス
検索
ディレクトリ
バッファ
セグメント番号
ページ番号
ページ枠番号

ページの多段構成


V 保護情報 M ページフレーム番号
PTE0
PTE1
PTEn
0 001100 1
1 101100 0
1 011100 1
ページ枠番号
ページ枠番号