- では、以降では、仮想メモリについて講義します。
  - キャッシュメモリについては、次回、講義します。

- 教科書の対応範囲は、以下の通りです。
  - 7.2節始め～7.2.3項(b)（p.246～254）

# 仮想メモリとは？

- ここでは、メインメモリに対する以下の要求事項について考えます。

1. メインメモリの容量を大きくしたい。
   - メインメモリに載り切らない大きなサイズのプログラムは実行できません。そのようなプログラムを実行するためには、例えば、データを分割して、今の計算に必要な部分をメモリに置き、それ以外の部分はファイルに保存します。メモリとファイルの間でデータを入れ替えながら計算を行うプログラムを書かなければなりませんが、とても面倒です。

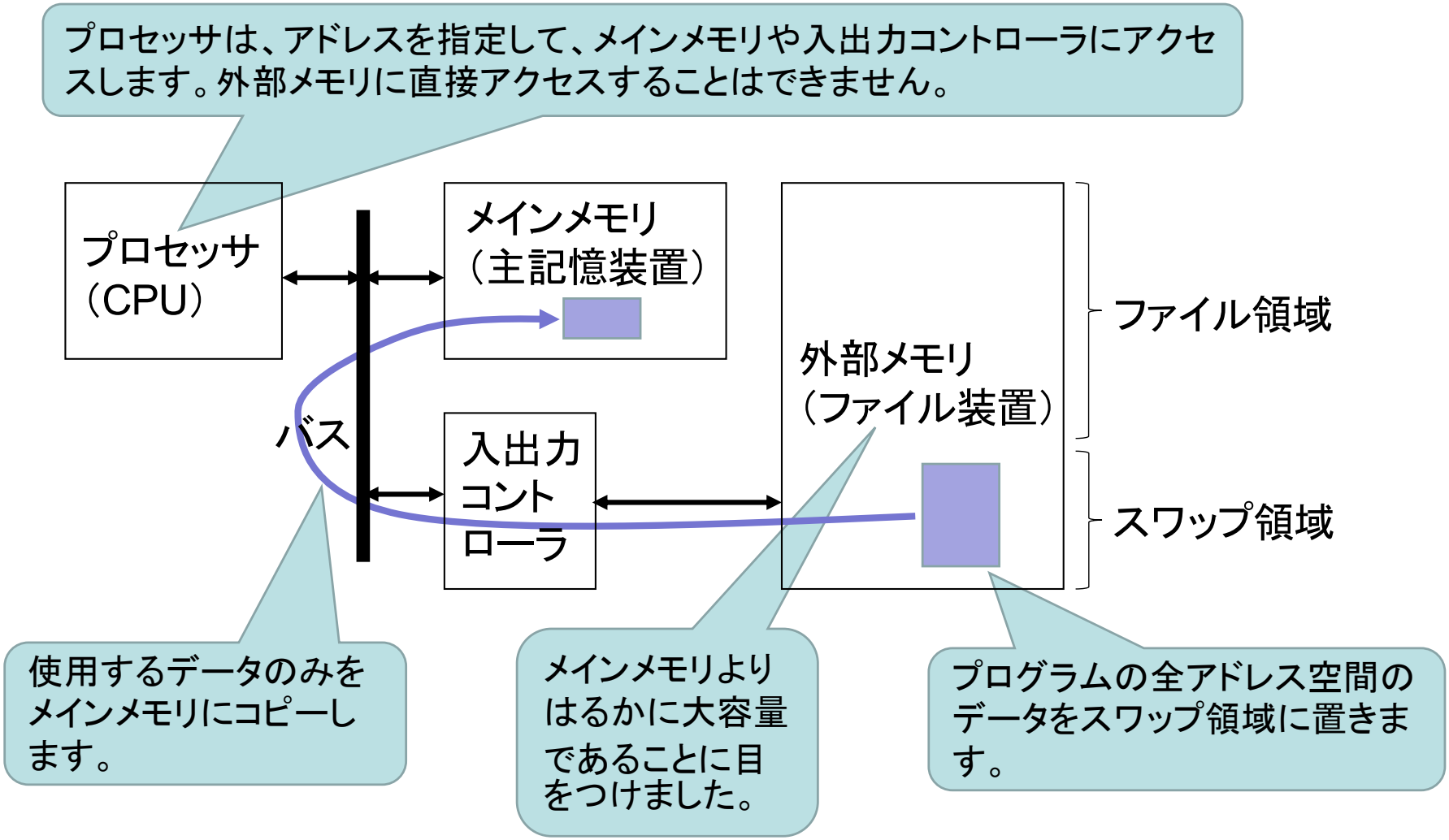
2. メインメモリの容量の違いによって、命令セットが異なってしまうことを防ぎたい。
   - 例えば、メインメモリの容量が1MBであれば、アドレスは20ビット必要です。256MBであればアドレスは28ビットです。アドレスのビット数が異なることで、命令セットも異なってしまうのは嬉しくありません。
     - 最初の授業で話しましたが、Amdahlが「アーキテクチャ」という言葉を使い始めたときのことを思い出してください。
   - そこで、アドレスを28ビットと決めてしまったら、ソフトウェア（プログラム）側に256MB使えるよ、と宣言していることになります。そのようなプログラムを、メインメモリが1MBのコンピュータでどうやって実行できるでしょうか？
   - 逆に、実際のメインメモリ容量以上のメインメモリ空間が提供できるのであれば、メインメモリ容量が256MBのコンピュータでも、アドレスを28ビットに制限する理由は無くなります。将来性（メインメモリ容量の増大）を考えれば、アドレスのビット数はもっと多くしておいてもよいかもしれません。
     - もちろん、命令語が長くなるのは嫌なのですが。

- かくして、命令セットで定義するメインメモリの容量（アドレスのビット数）は、実際にコンピュータに搭載されているメインメモリの容量とは独立に定義されることとなったのです。
  - 命令セットで定義した（容量の）メインメモリ（空間）を仮想（論理）メモリと呼び、マシン命令で指定するアドレスを仮想（論理）アドレス（virtual/logical address）といいます。
  - コンピュータに搭載された実際のメインメモリを実（物理）メモリと呼び、そのアドレスを実（物理）アドレス（real/physical address）といいます。

    注） 仮想アドレスと論理アドレス、実アドレスと物理アドレスが厳密には異なるアーキテクチャもあるが、だいたいは、同じであると考えてよい。

- プログラム内で使用するアドレスは仮想アドレスです。
  - 「プログラム内で使用するアドレス」は「マシン命令で指定するアドレス」ですから、当然、仮想アドレスです。
  - アドレッシングモードに従って実効アドレスを求めますが、この実効アドレスも仮想アドレスです。
    - アドレッシングモードの講義では実メモリのように説明していましたが、方便です。
  - 仮想メモリの機能や実現について色々な方式が考案されましたが、現在は多重仮想アドレス方式（MVS; Multiple Virtual Storage）で落ち着いています。
    - MVSでは、プロセス毎に、1つの独立した仮想アドレス空間を与えます。
      - 例えば、アドレスが32ビットのシステムでは、1つのプロセス（例えば、1+1=2を計算するだけのプログラムであっても）に、4GBの仮想アドレス空間が与えられます。

# 仮想メモリの原理


プロセッサは、アドレスを指定して、メインメモリや入出力コントローラにアクセスします。外部メモリに直接アクセスすることはできません。
プロセッサ
（CPU）
メインメモリ
（主記憶装置）
バス
入出力
コント
ローラ
外部メモリ
（ファイル装置）
ファイル領域
スワップ領域
使用するデータのみをメインメモリにコピーします。
メインメモリよりはるかに大容量であることに目をつけました。
プログラムの全アドレス空間のデータをスワップ領域に置きます。

- 仮想メモリ空間内の情報（データ/命令語）は、ファイル装置に格納します。
  - ファイル格納用の領域以外に、スワップ（swap）領域を確保して、ここにすべての仮想メモリ空間の情報を格納します。
    - OSによっては、スワップ領域全体を、1つの隠しファイルとして見せているものもあります。

- 仮想メモリ空間内の必要な情報のみをメインメモリ（実メモリ）上に置きます。
  - 当然、プロセッサはファイル装置内のスワップ領域に、直接、アクセスすることはできません。直接、アクセスできるのは、メインメモリのみです。
  - 仮想メモリ空間内の必要な情報が、メインメモリ（実メモリ）に格納されているか否か、また、格納されているなら実メモリのどこに格納されているか、を管理しなければなりません。
  - また、メインメモリにアクセスする際には、実アドレスを用いてアクセスしなければなりません。従って、仮想アドレスを実アドレスに変換する必要があります。

- 必要な情報が実メモリに存在しない場合は、OSがスワップ領域から読み込んで実メモリに配置します。
  - このとき、実メモリに空きがなければ、不要と判断させる情報をスワップ領域に書き戻します。これをスワッピング（swapping）といいます。

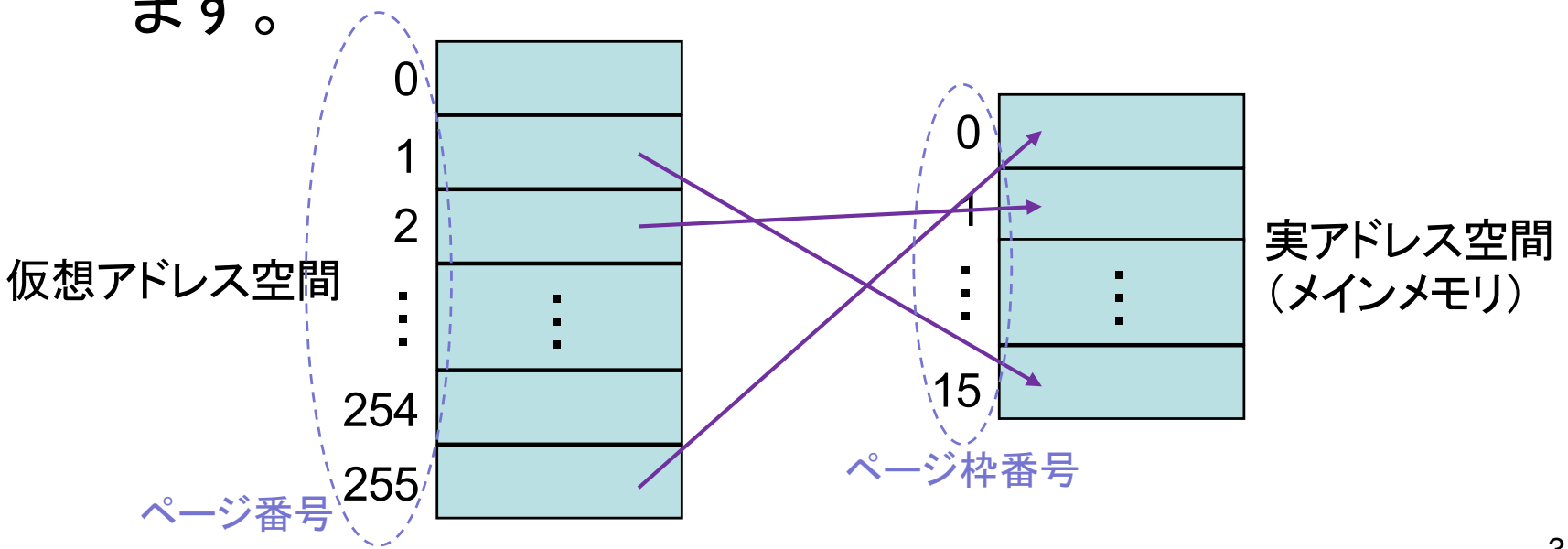
# アドレスマッピング

- メインメモリとスワップ領域との間のデータ転送は、入出力の一種です。プロセッサやメインメモリに比べてはるかに遅いファイル装置を相手に、少量のデータを転送していたのでは効率が悪くなります。従って、1回にある程度の大量のデータを転送します。
- そこで、その転送単位と情報の管理方法として、以下の2つの方式があります。
  - ページング（Paging）方式
    - 固定長の領域を単位として転送および管理する。
  - セグメンテーション（Segmentation）方式
    - 可変長の領域を単位として転送および管理する。

- 論理的・機能的には、情報の意味を考えて領域分割し、その結果の可変長領域を単位として仮想アドレスと実アドレスの対応関係（マッピング）を管理するセグメンテーション方式の方が望ましいのですが、管理が複雑になります。
- 一方、ページング方式では、記憶内容とは無関係にサイズのみで領域分割を行うので、管理は簡単になります。
- 実際には、セグメンテーション方式とページング方式とを組み合わせた方法で仮想メモリを実装していますが、本講義では、ページング方式についてのみ述べることにします。

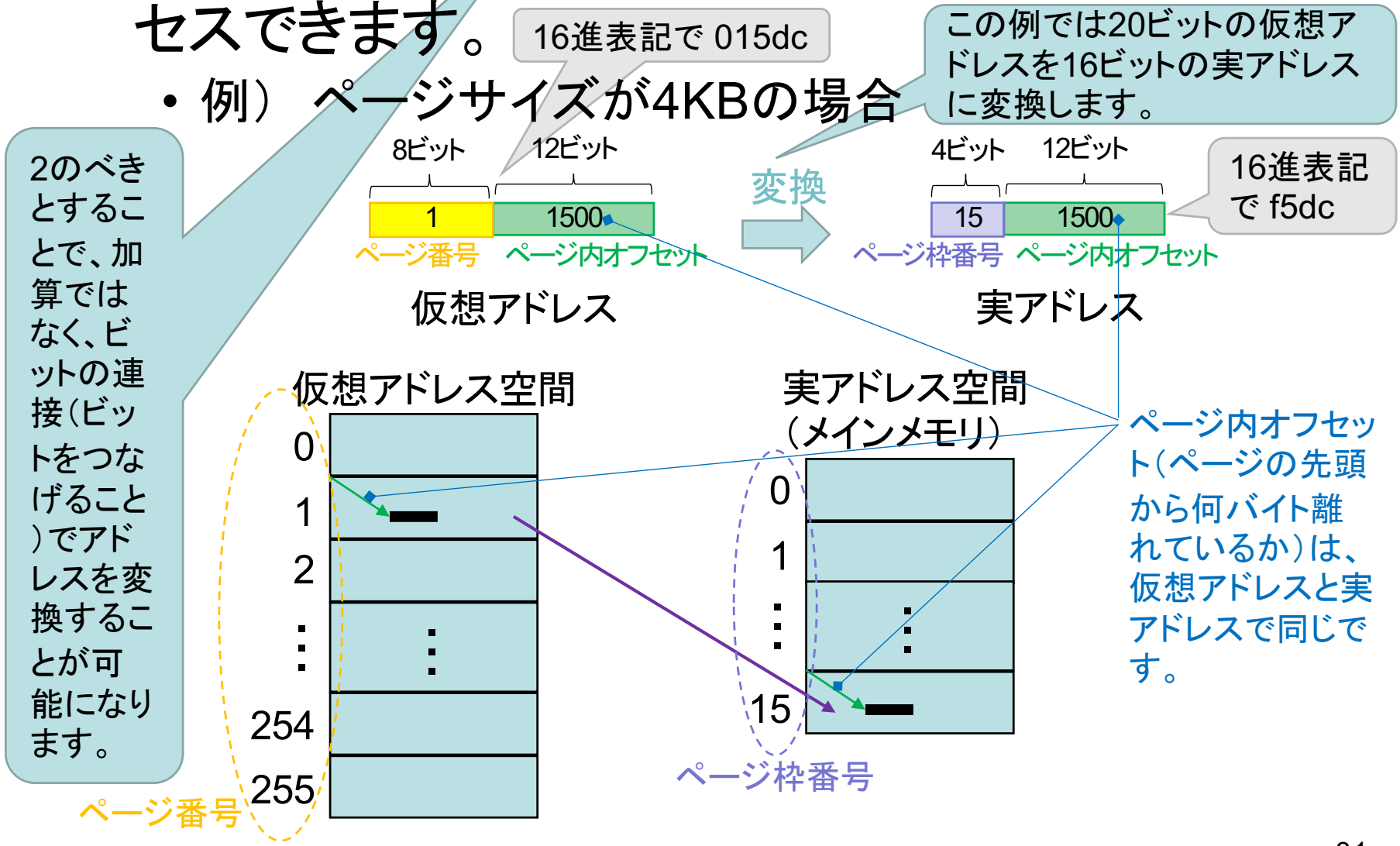
コンピュータシステムというよりは、OSの内容です。

- ページング方式でのアドレスマッピング
  - 仮想アドレス空間を、固定サイズのページに分割し、順にページ番号を振ります。
  - メインメモリ（実アドレス空間）の領域も同じサイズのページに分割し、順にページ枠番号を振ります。
  - 仮想アドレス空間の何番目のページが、メインメモリの何番目のページ枠に格納されているか、を管理します。



– ページサイズを2のべきにすると、ページ番号をページ枠番号に変換するだけで、実メモリにアクセスできます。

- 例） ページサイズが4KBの場合



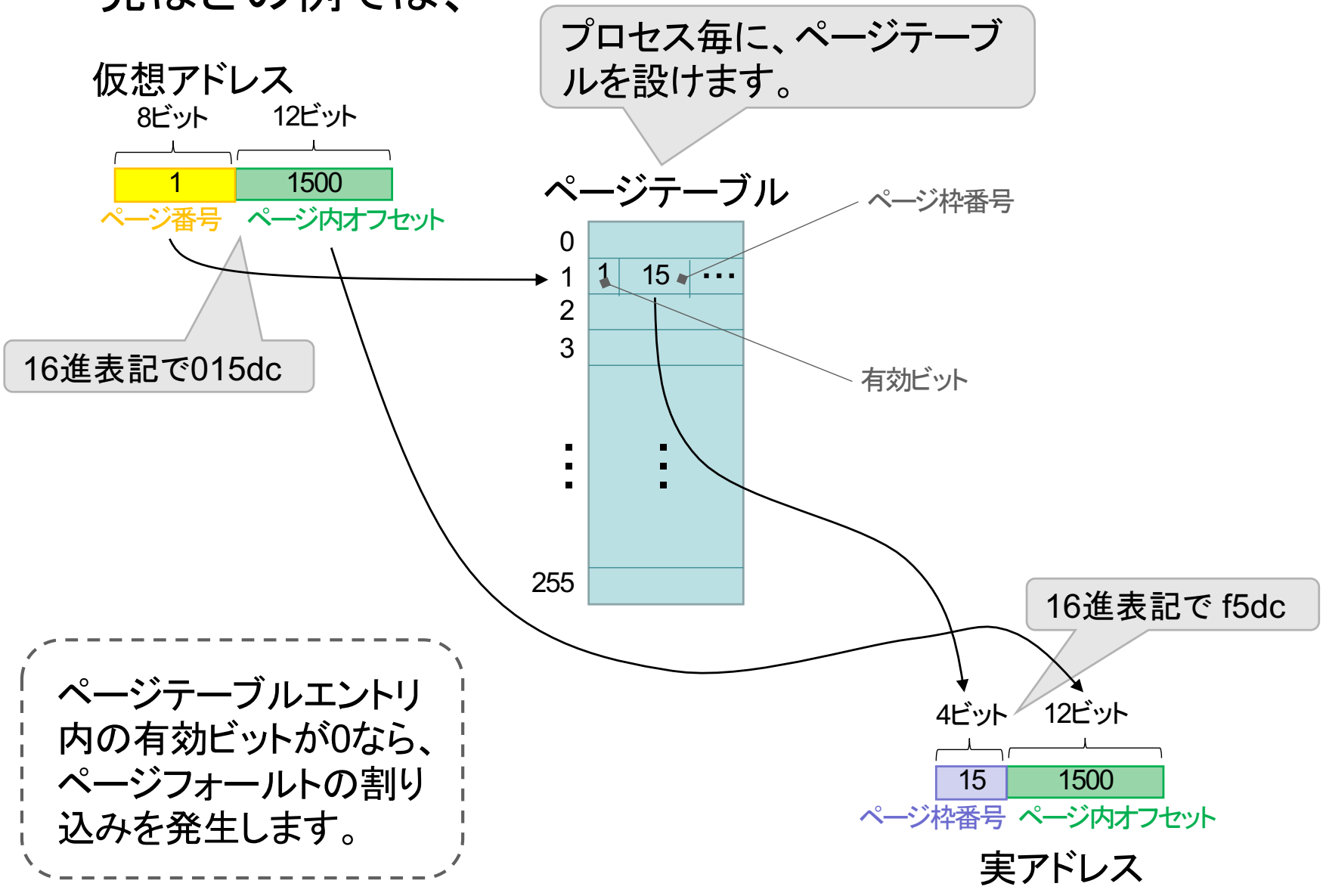
# 動的アドレス変換(DAT)機構

- 前ページのアドレス変換を、プロセッサがメインメモリにアクセスするたびに行います。その仕組みを、動的アドレス変換(DAT; Dynamic Address Translation)機構といいます。
  - ページ番号とページ枠番号の対応は、ページテーブルと呼ぶ表を用いて管理します。
    - ページ番号をインデックスとして、ページテーブルのエントリを参照します。各エントリには、以下の情報を保存します。
      - 有効ビット: このページがメインメモリにあるかどうかを示す。
      - メインメモリに存在する場合のページ枠番号
      - メインメモリに存在しない場合の、スワップ領域内でのアドレス情報

      など。

構造体の1次元配列のようなものだと考えればよいでしょう。その配列の添字(インデックス)に、ページ番号を指定します。

– 先ほどの例では、


プロセス毎に、ページテーブルを設けます。
仮想アドレス
8ビット
12ビット
1
1500
ページ番号
ページ内オフセット
16進表記で015dc
ページテーブル
ページ枠番号
有効ビット
0
1
2
3
255
1
15
16進表記で f5dc
4ビット
12ビット
15
1500
ページ枠番号
ページ内オフセット
実アドレス
ページテーブルエントリ内の有効ビットが0なら、ページフォールトの割り込みを発生します。

- メモリ管理、プロセス管理、入出力管理は、本来、OSが行うべき仕事です。
  - これらの仕事はハードウェアで行うには複雑過ぎて、また、柔軟性を欠きます。
- しかし、メモリにアクセスするたびに、OSを呼び出していたのでは、性能がガタ落ちです。
- そこで、ある程度はハードウェアで行い、ややこしい処理が必要になれば、割り込みを起こしてOSを呼び出します。例えば、
  - 最近変換した情報（ページ番号とページ枠番号の対）をハードウェアの連想記憶によって検索し、瞬時に実アドレスを求めます。この機構をアドレス変換バッファ（TLB: Translation Look-aside Buffer）といいます。検索して見つからない場合はTLBミスの割り込みを発生します。
  - ページテーブル自体はOSで管理するべきものですが、そのページテーブルを読み出して自動的にアドレス変換を行うハードウェアを備えているコンピュータもあります。ただし、それを用いるためには、その機構で想定しているページテーブルのデータ構造に合わせて、OSを作らなければなりません。

# おまけ

- プログラマにとって、データ参照の局所性を高めることは、プログラムの高速化に繋がります。
- あちこちのページにアクセスするようなプログラムを書くと、頻発するページフォールトによってOSばかりが動くことになり、プログラムの実行時間が長くなってしまいます。
- コンピュータシステム、コンピュータハードウェアの設計者のみならず、ソフトウェア設計者にとっても、コンピュータハードウェア（コンピュータシステム）がどのような仕組みで動いているのかを知ることは重要なことなのです。