オペレーティングシステム

2022/12/13

海谷 治彦

目次

- 仮想メモリ

- 1 概要
- 2 利点
- 3 アドレス変換
- 4 ページング
- 5 メモリスケジューリング
- 6 性能

問題提起

- プロセスは相互に他プロセスのメモリに原則アクセスするとマズい.
- とはいえ，物理的なメモリ自体は，共用せざるを得ない.

- プロセス毎に使うメモリ空間を独立させるにはどうすればよいか？

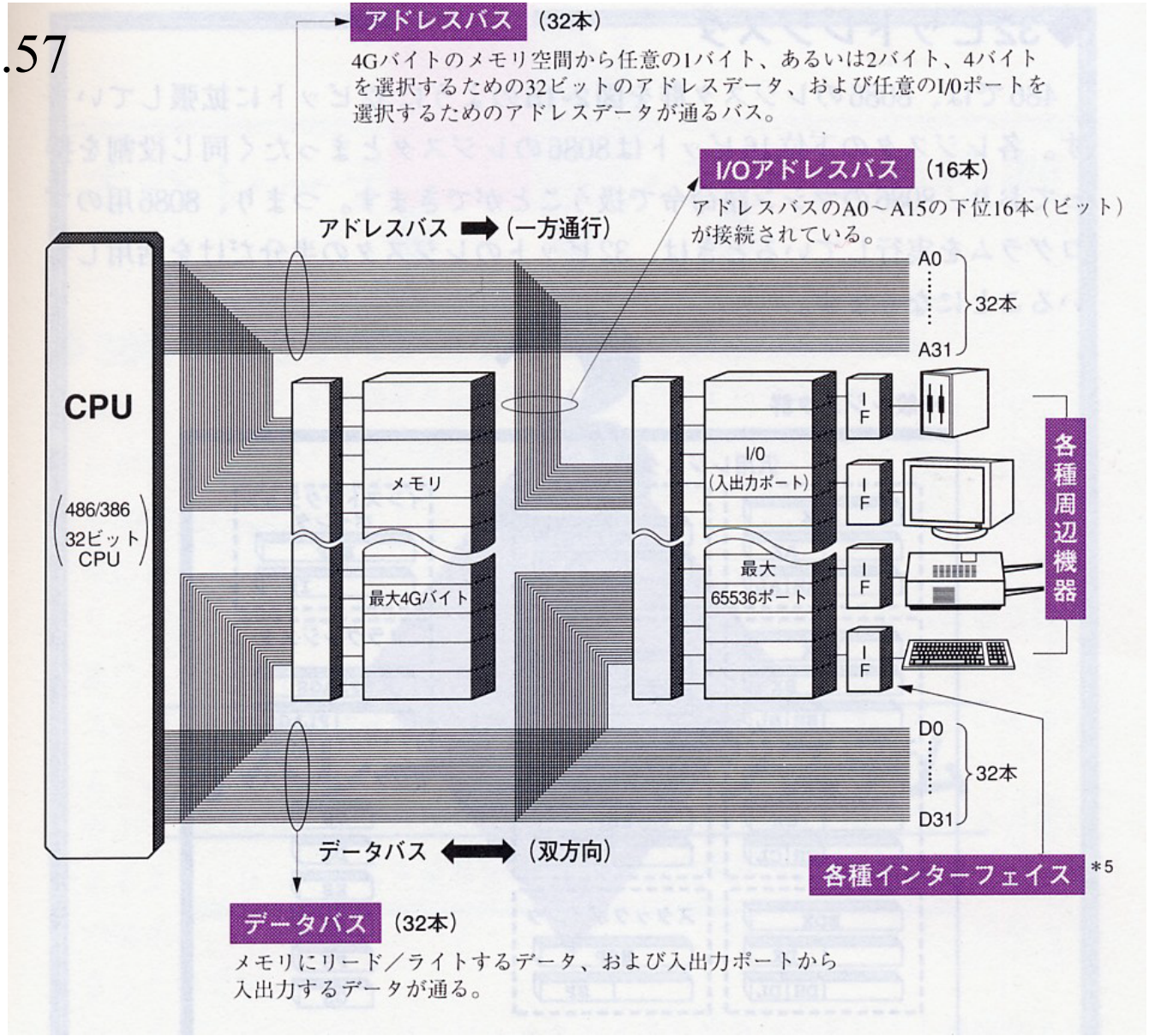
- OSが扱えるアドレスの広さ(メモリ量)はCPUのアドレスバスの長さに依存する.
- しかし，実際にマシンに搭載されているメモリ量は持ち主の財布具合に依存する.

- 実際は少ない実メモリを使って，どのようにOSが扱えるアドレス全てを使えるようにするか？

復習

実際の386CPUの構造

文献 p.57



復習

参考: アドレスバス本数とメモリの広さ

$2^{16} = 65536 \fallingdotseq 65K$

$2^{32} = 4294967296 \fallingdotseq 4G$

$2^{64} = 18446744073709551616$

$\fallingdotseq 18E$ (エクサ)

32bitマシン主流の時代,
4G以上のメモリが無駄
といわれた所以は上記にある.

参考: バス本数が3本の場合
(2^3=8) 8個のアドレスしか
指定(区別)できない.

1本	2本	**3本**
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

問題提起に対する解決案

- アドレス変換によって，プロセス毎の利用する実アドレスを原理的に分離しよう．
- ※ ほぼ4回目の復習になります．

- 仮想記憶によるアドレス空間の拡大．
- 使ってないメモリの中身をディスク等に退避する．
  - 先週のスワップに同じ．
- ※ 双方を仮想記憶と呼ぶ場合がある．

復習

i386の3つのアドレス記述法

- 論理アドレス: マシン語でのアドレス指定に使う形式．セグメントとオフセットの対で表現．
- リニアアドレス: 4GBのメモリ空間を素直に表現した形式．32ビット
- 物理アドレス: メモリチップの実アドレス．表現は32ビットだが，上限は実際に搭載しているメモリ量に依存．

Linuxの場合，4Gサイズのセグメントを使うので，

論理アドレス = リニアアドレス

文献5 p.44

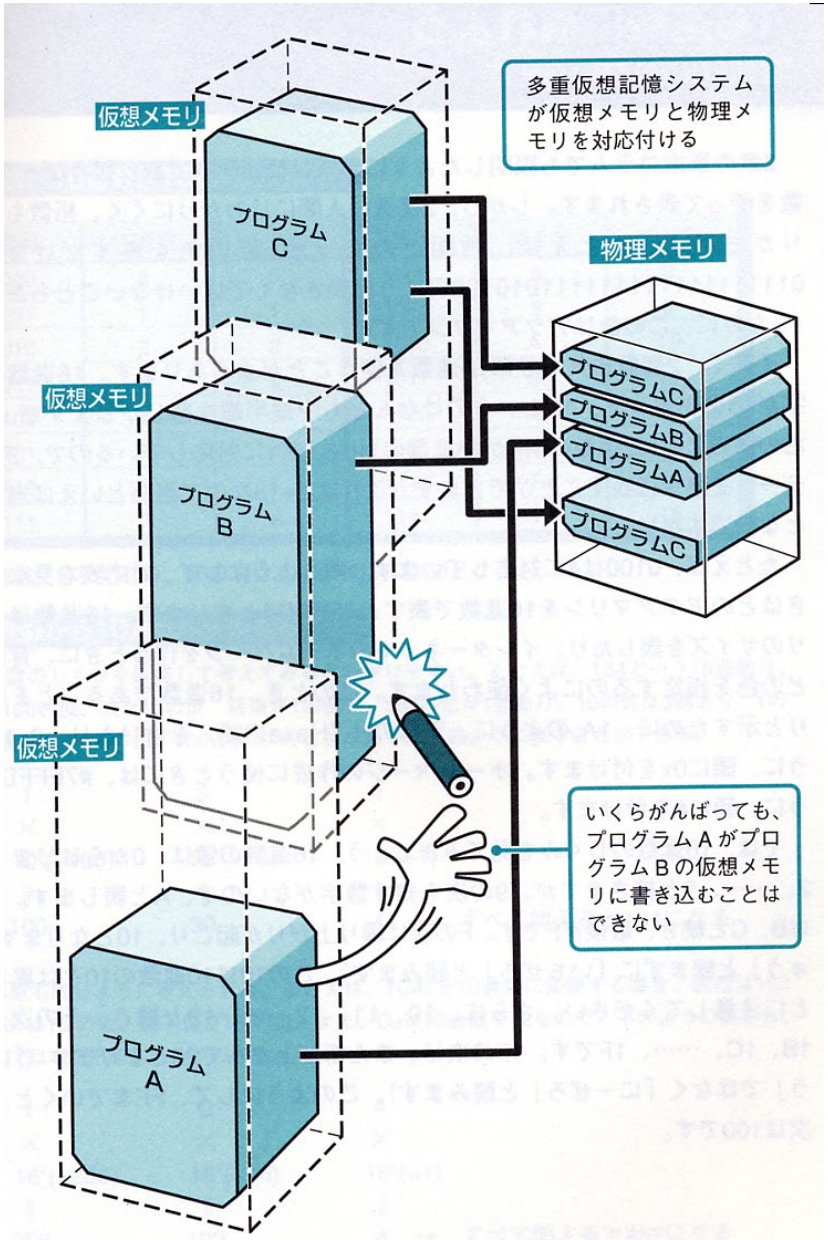
アドレス変換について

復習

多数プロセスの交通整理: メモリ

- Linuxではアプリ用のセグメント(USER_CS, USER_DS)は一対しかない.
- ご存じの通り100個近いプロセスが同時に存在するのが普通.
- 異なるプロセス同士が同じメモリを使ったら混乱が生じるのは前回話した通り.
- Linuxではアドレス変換テーブルを切り替えることで，複数のプロセスが同じアドレスをアクセスしても，実際には違うメモリ回路を使うように工夫されている.

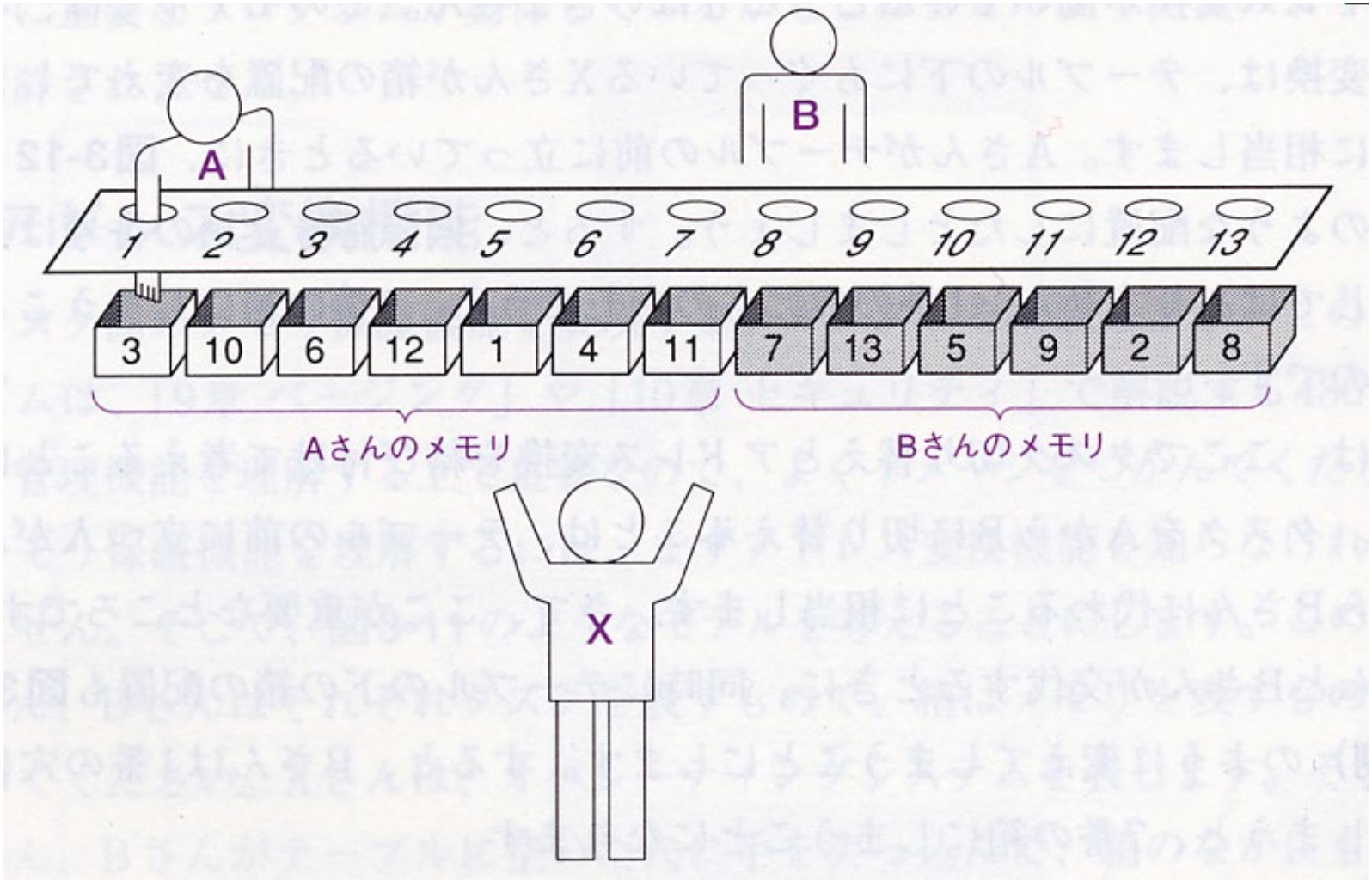
概念図


多重仮想記憶システムが仮想メモリと物理メモリを対応付ける
仮想メモリ
プログラムC
物理メモリ
プログラムC
プログラムB
プログラムA
プログラムC
仮想メモリ
プログラムB
仮想メモリ
いくらがんばっても、プログラムAがプログラムBの仮想メモリに書き込むことはできない
プログラムA

文献2 p.95

復習

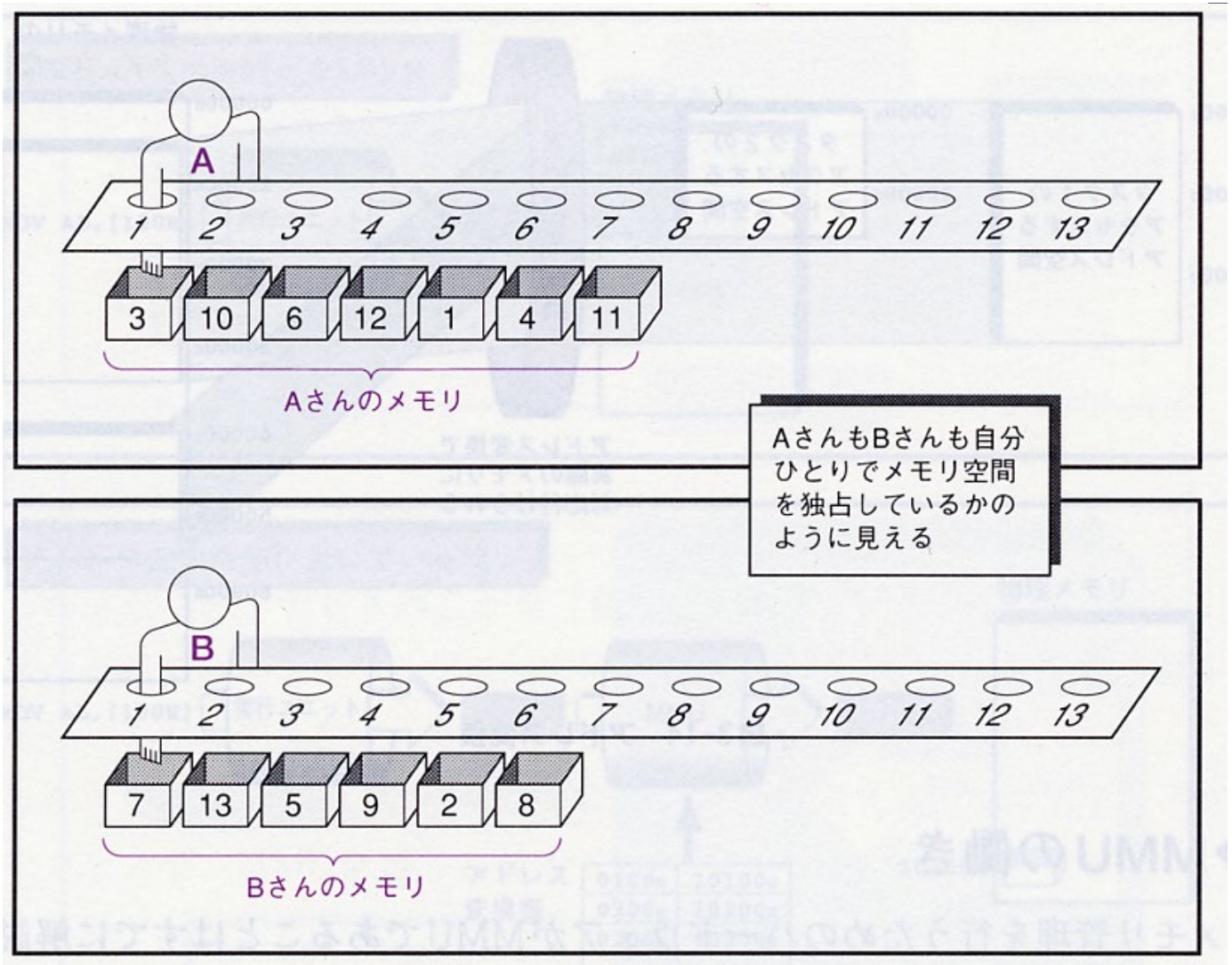
アドレス変換の考え方 1



文献6 p.79

復習

アドレス変換の考え方 2



復習+

ページング

- 0～4GBで表現されるリニアアドレスを物理アドレス(実際にマシンに搭載されたアドレス)に変換する仕組み.
- 4GBも実際積んでないマシンでも4Gあるように振る舞えるのはこの機構のおかげ.
  - 64bitアドレスならなおさら,
  - 18E Bのメモリなど搭載するのは無理.
- もし4GBメモリのマシンがあったとしても, i386上で, 複数のプロセスが同時に動作するOSを稼動させるには必要な技術.

復習

ページ

- セグメントを4KB(もしくは4MB)の固定長に分割した個々の部分のこと.
- リニアアドレスのある部分が物理アドレスのどの部分に対応するかは表で管理している.

  ⇒ アドレス変換テーブル

セグメントとページ

- メモリは1B単位のデータが順に並んで，それに番地(アドレス)がついているだけである.
- 効率の観点から1B毎に，ちまちま扱うのではなく，ある程度のグループ化をする.

セグメント: 任意の大きさに区分けする.

ページ: 固定の大きさに区分けする.

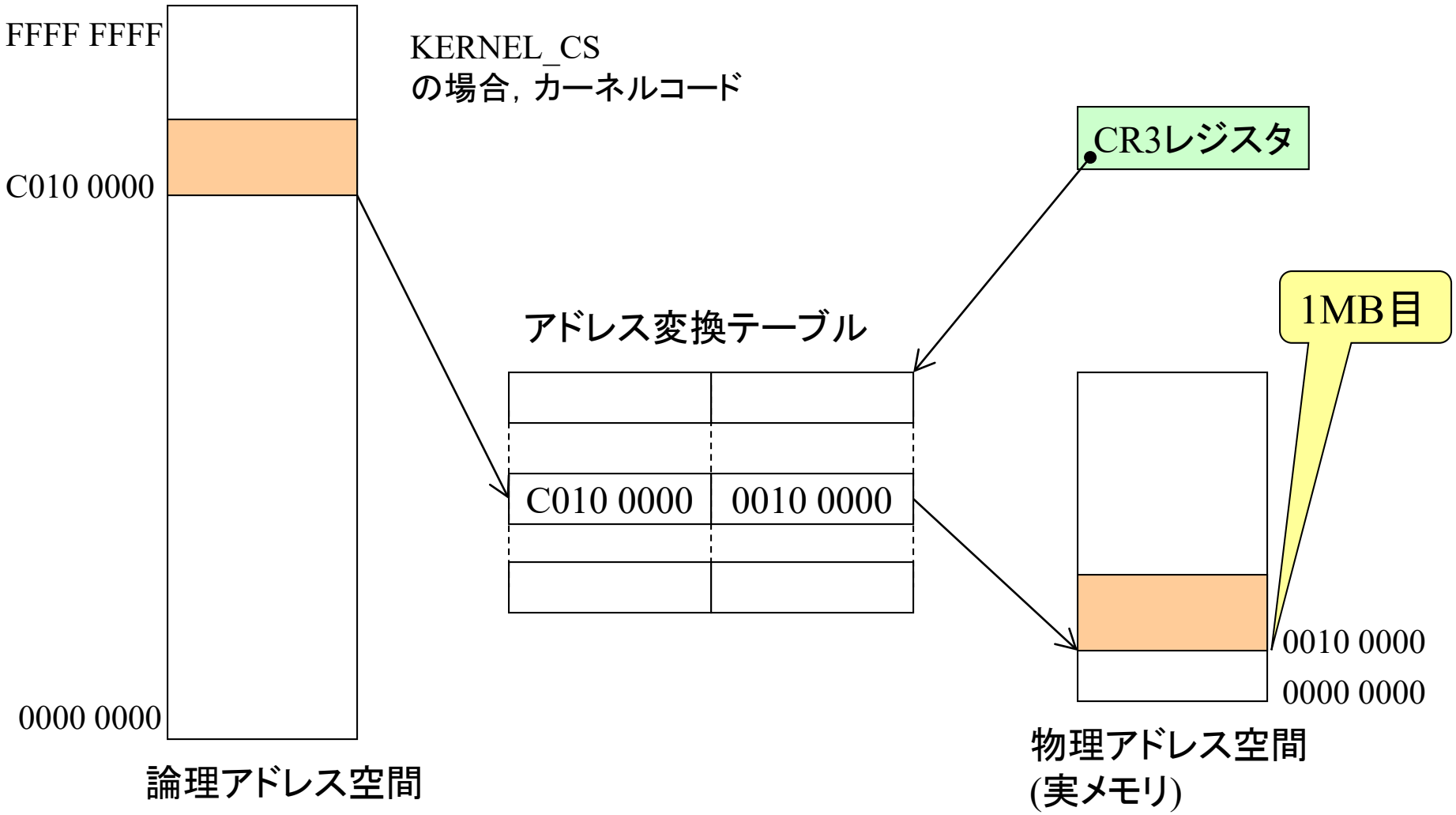
復習

アドレス変換テーブル

- 前述のようにリニアアドレスのある部分が物理アドレスのどの部分に対応するかを記述した表.
- 無論，メモリ内に記入される.
- ある瞬間に利用されているテーブルは1つだが，変更もできる.
- i386ではCR3レジスタにこのテーブルがあるアドレスが記入されている.

復習

アドレス変換テーブルの例



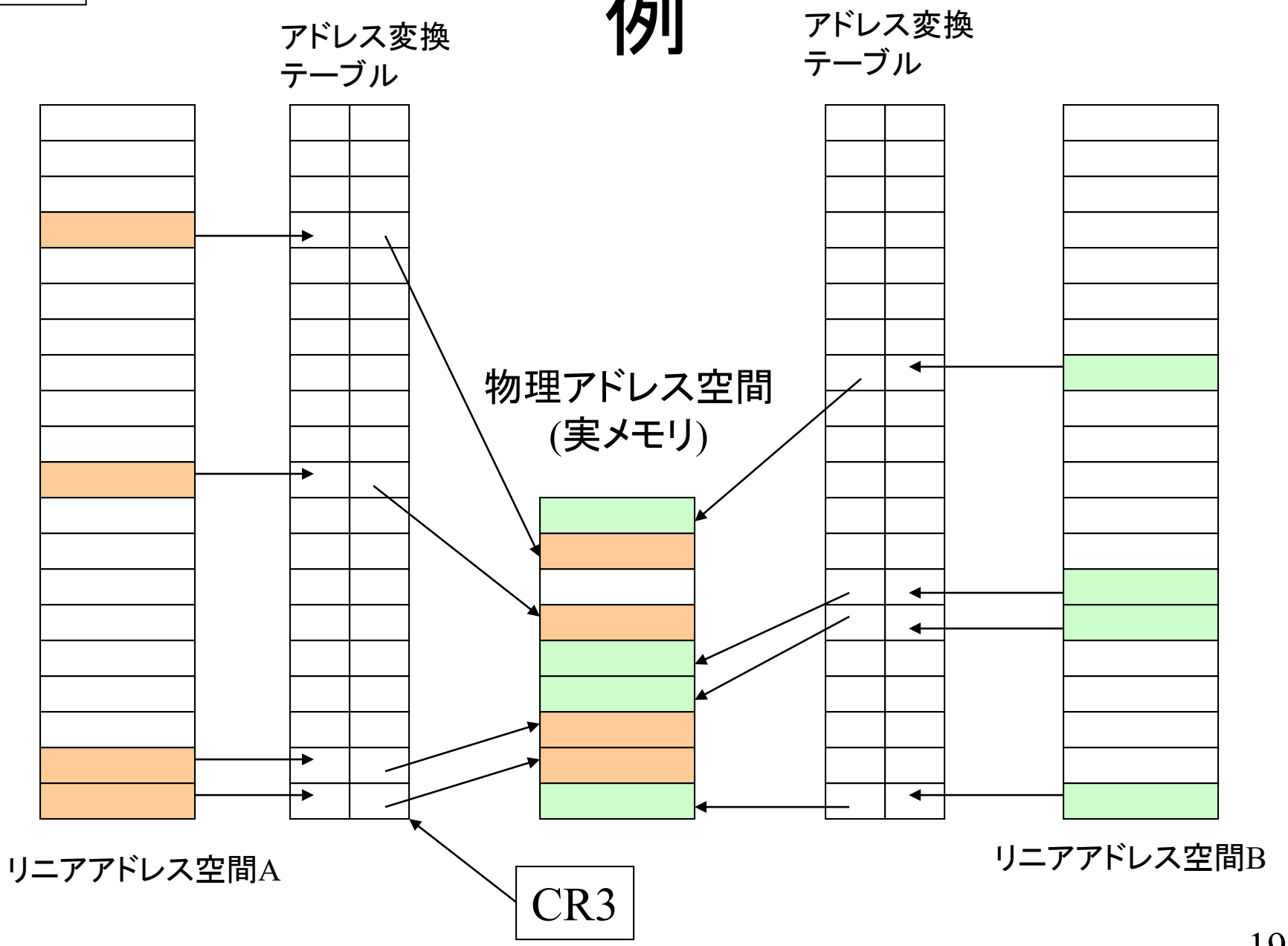
復習

アドレス変換テーブルとプロセス

- Linuxではプロセス毎にアドレス変換テーブルを作る.
- さらに，プロセス間で同じ物理アドレスを使わないようにテーブル内の値をOSが設定する.
  - 意図的に共有させることもできる.
- ユーザープロセスがこのテーブル群やCR3の内容は変更できないので安心.
  - テーブル群はKERNEL_DSに書かれる.
  - CR3はKERNEL_CS内のコードでしか変更できない.
- 実際のテーブルは効率化のため，二段階テーブルになっているが，それについては後日に.

復習

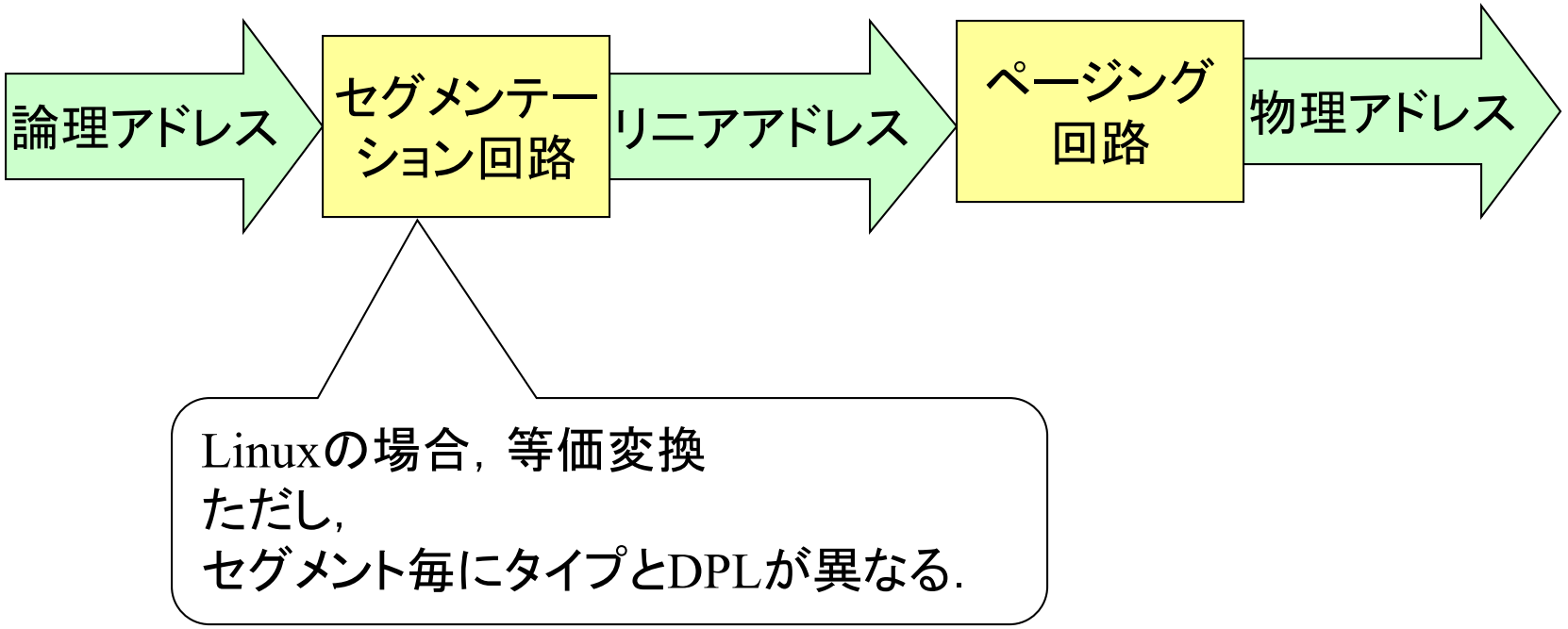
例


アドレス変換
テーブル
アドレス変換
テーブル
物理アドレス空間
(実メモリ)
リニアアドレス空間A
リニアアドレス空間B
CR3

復習

アドレス変換

マシン語が解釈されて，実際のメモリ回路までたどり着くには，以下のような二段階の変換が行われる．



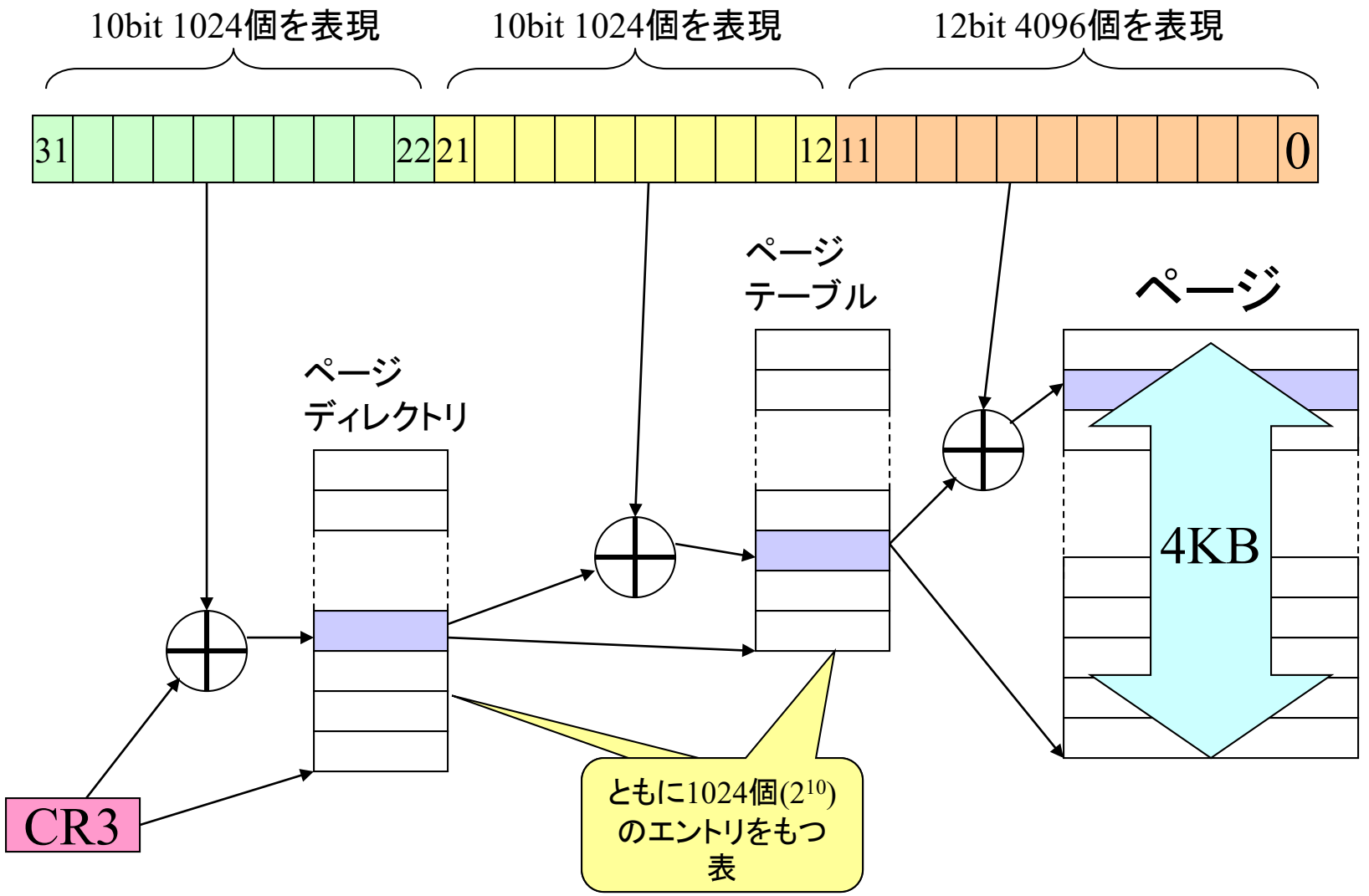
i386のページ

- 通常，1個のページは4KB (2^{12}=4096B)の大きさ.
- よって，4G(=2^{32})のメモリ空間は，およそ100万個(2^{20}個)のページに分けられる.
- 前述の仮想記憶は，このページ単位にメモリの置き換えを行う.

ページング回路について

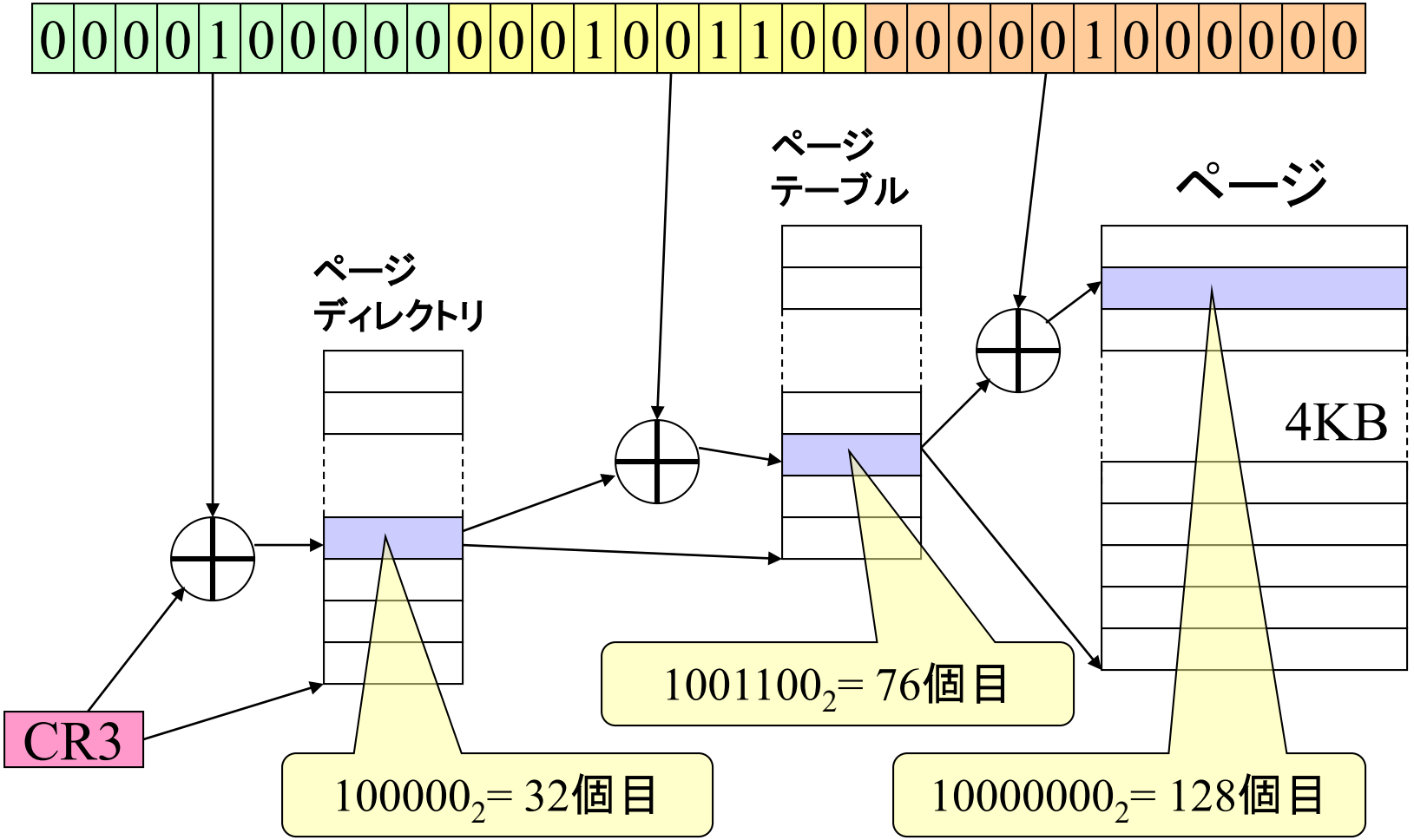
- 100万個(2^{20}個)の表で検索するのは手間だし，そもそも表自体のサイズが大きすぎる.
- そこで，ページを3階層の木構造で表現している.
- これによって表を記録するサイズと，検索の時間の面から効率が良くなっている.

ページング回路 (4KB用)



例

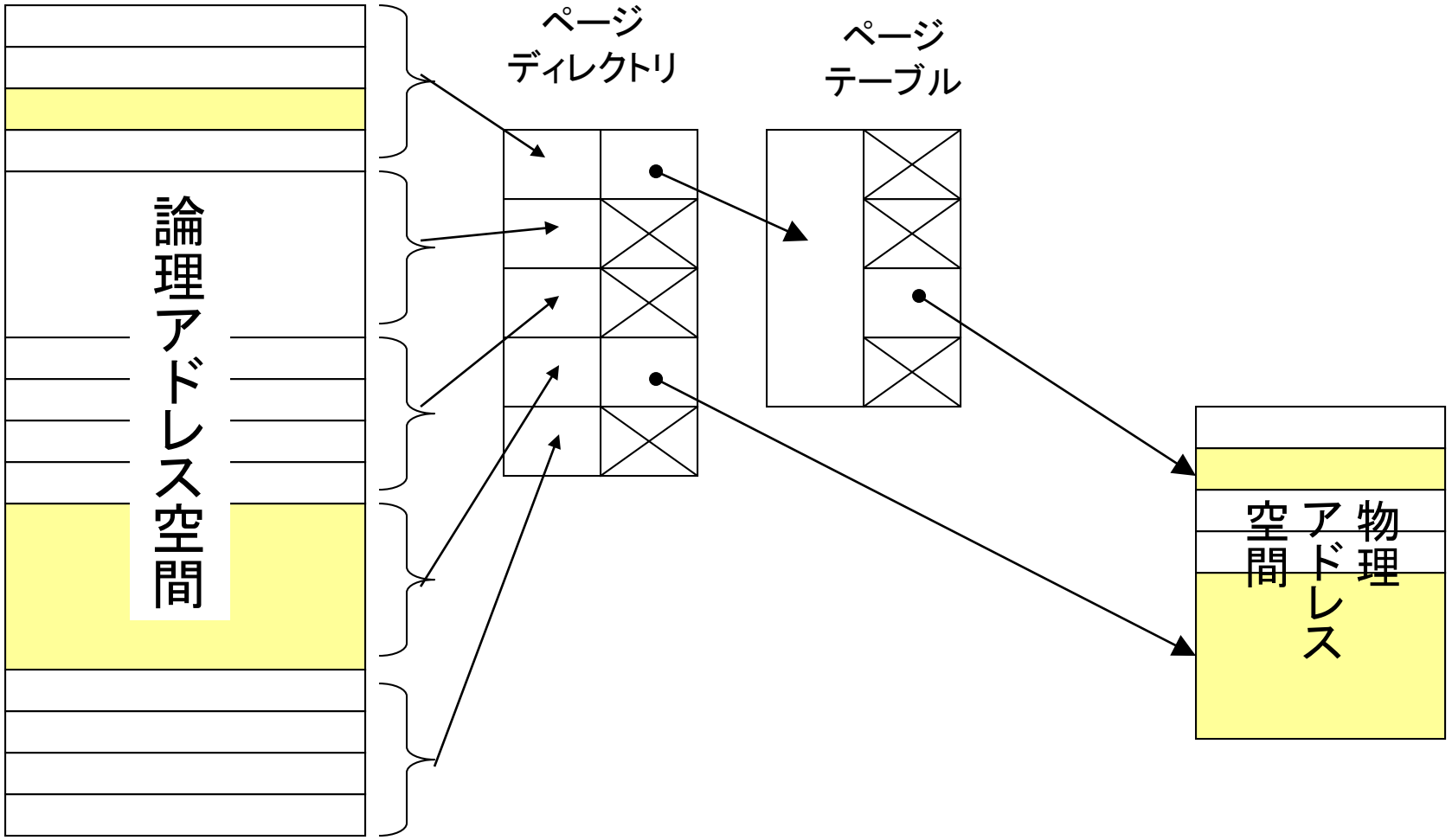
仮想アドレス 0804 c040 =



4MBのページ

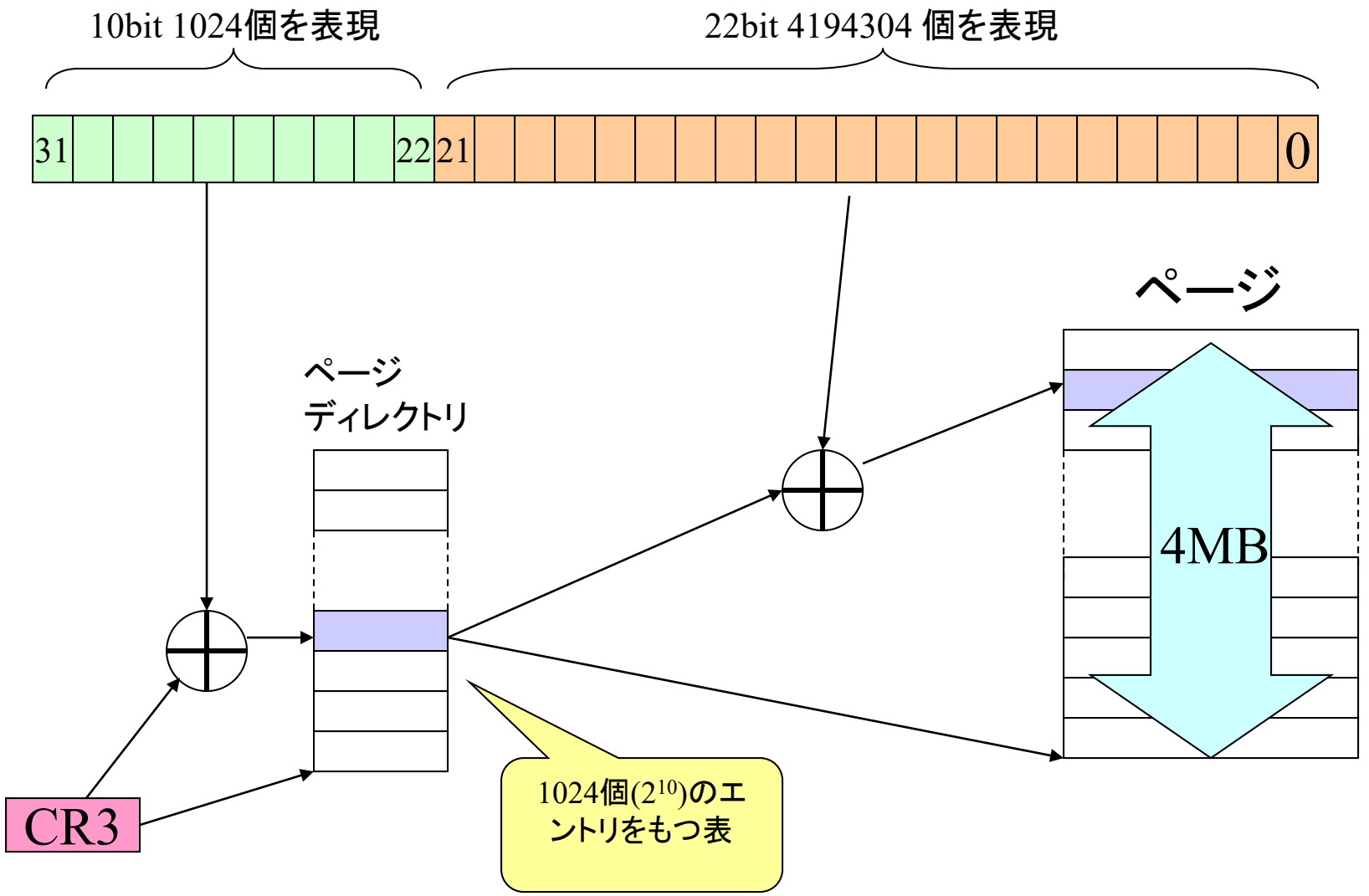
- Pentium以降では，4MB(=2^{22})のページを扱えるようになった.
  - Pentium: i386と現在のCPUとの中間の世代のCPU. 1990年代に主流.
    - 今のAMDのヤツじゃないです.
- しかも，4KBのページと混在して扱っても良いことになった.
- 4MBのページを扱うか否かは，CR4レジスタ内のとあるビットで決まる.

4Kと4Mのページ混在イメージ



※ 4Mと4Kでは1/1000の違いがあるので，上図は実際のサイズ比を反映してません．

ページング回路 (4MB用)



リアルな変換表の例

- 授業ページから参照のこと
- このデータを得るプログラムは古いカーネル(Kernel2.4)でしか動かない.
- おそらく, ウエブサーバーのプロセス

- 4Kページ 2191個 4Mページ 64個
- 結果, およそ265MBのメモリをこのプロセスは使っている.
  - 4GBのアドレス空間全体からしたら6%程度.
- うち, 16個の4Kページがスワップアウトしている. 0.002%以下のスワップアウト率.

カーネルから見たメモリ

- 物理メモリの1MB目から，2MB程度，カーネルのコードとデータが置かれる.
- 最初の1MBは，BIOSが利用するため，Linuxでは使用しない.
- カーネルコードとデータはスワップアウトされない.
- ページテーブル等はカーネル部分にある.

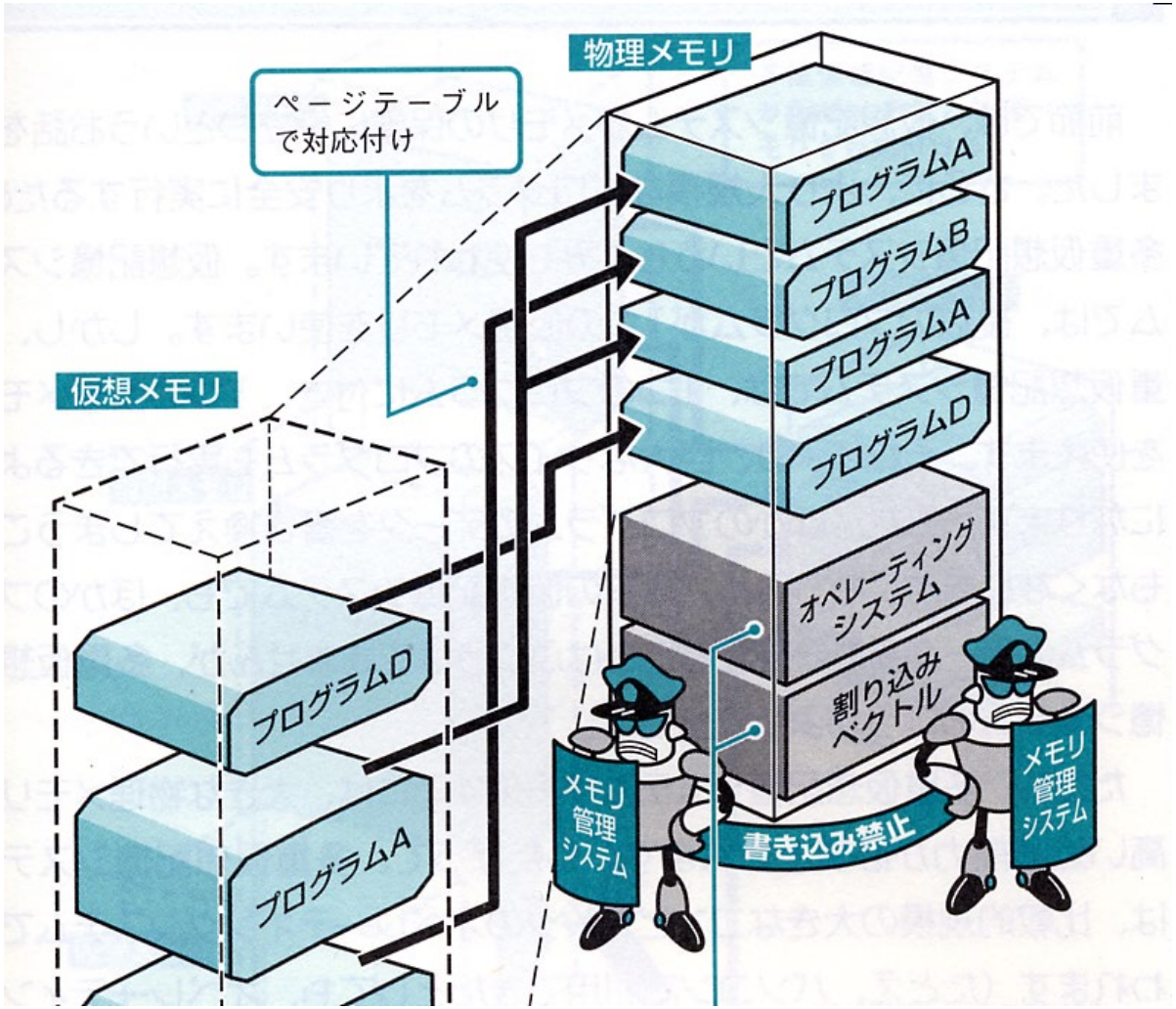
BIOS (ばいおす)

- Basic Input Output System の略
- ハードウェアのチェックをし,
- ブートセクタと呼ばれる記憶装置の特定位置にあるプログラム(通常はOSの起動準備プログラム)等の実行を開始する機能を持つ.
- BIOSが故障する(飛ばす)とマシン自体が全く起動しなくなる(涙)
- 最近は, UEFI ってのが代わりに使われる場合も多い.

一つのプロセスから見たメモリ

- 4Gのサイズを持つメモリに見える.
- これをLinuxでは仮想アドレス空間と呼ぶ.
- 4Gの内訳は,
  - 最初の3G ユーザーセグメント
    - プロセスのコードとデータがおかれる.
  - 残りの1G カーネルセグメント
    - 前述のカーネルがおかれる物理メモリの一部にマップされる.
      - こうしないとプロセスからカーネルが見えなくなり, システムコールが呼べなくなる.

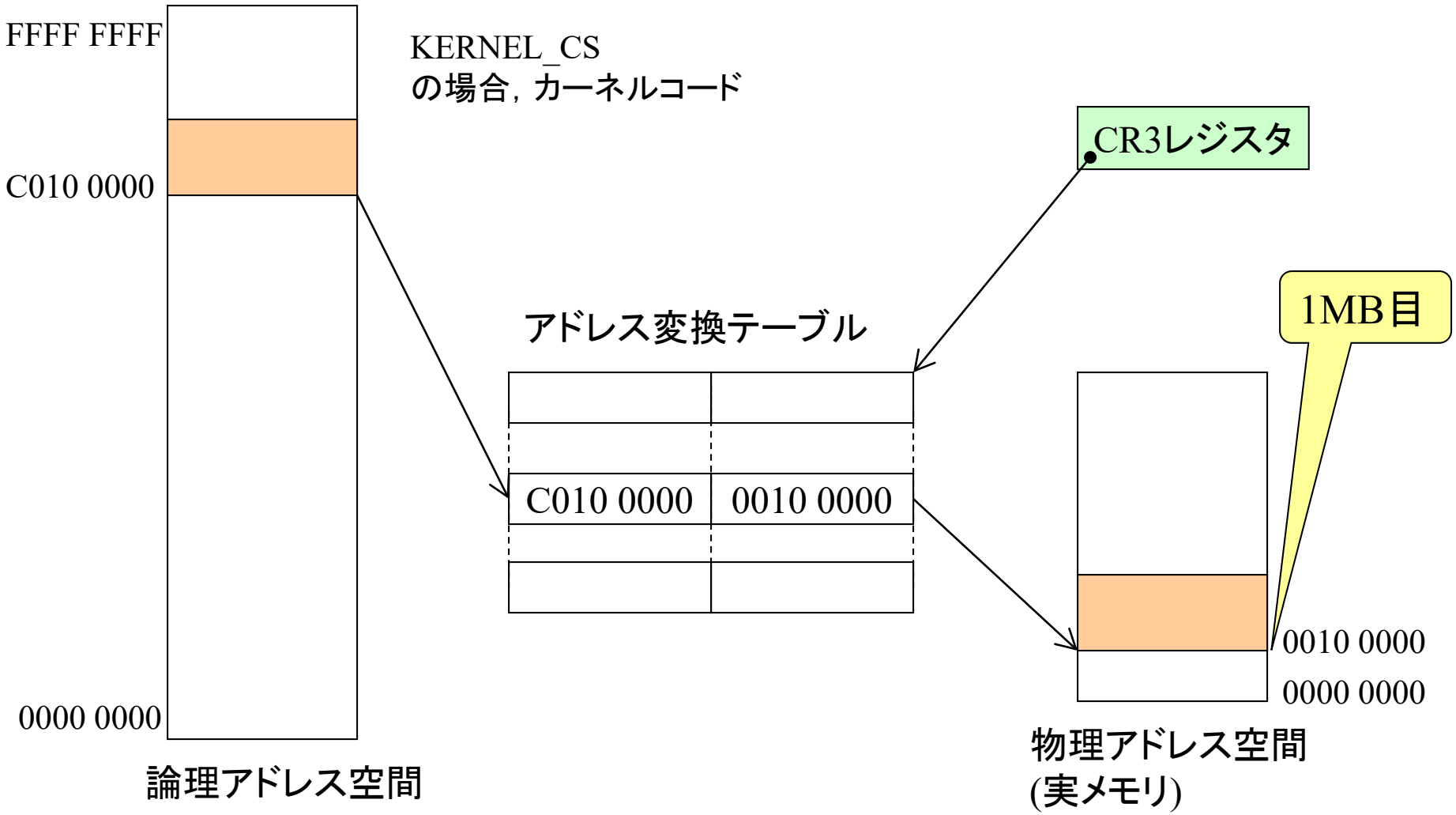
物理メモリの概念図



文献2
p.93

再掲載

アドレス変換テーブルの例

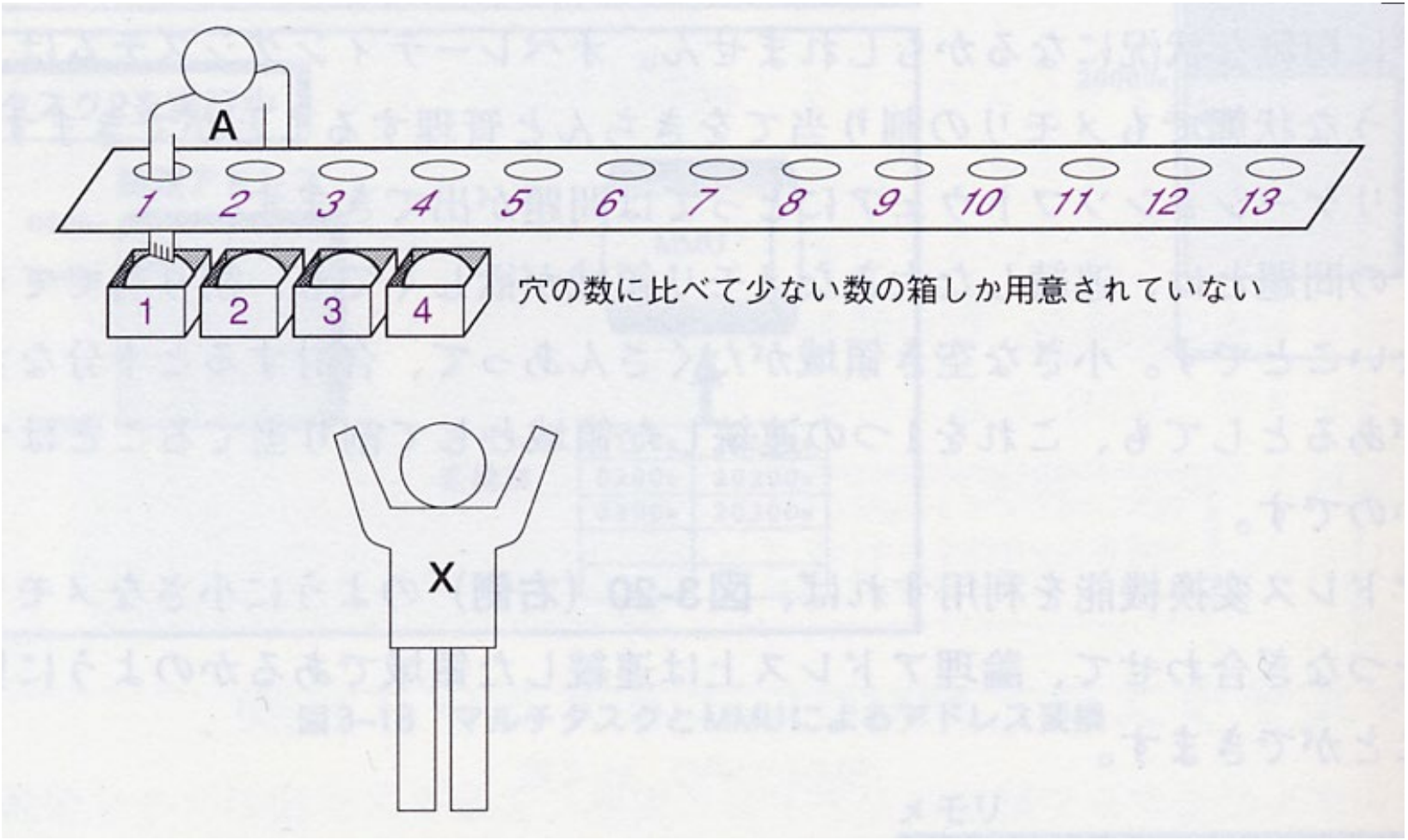


仮想記憶

少ない実メモリ上で大メモリを

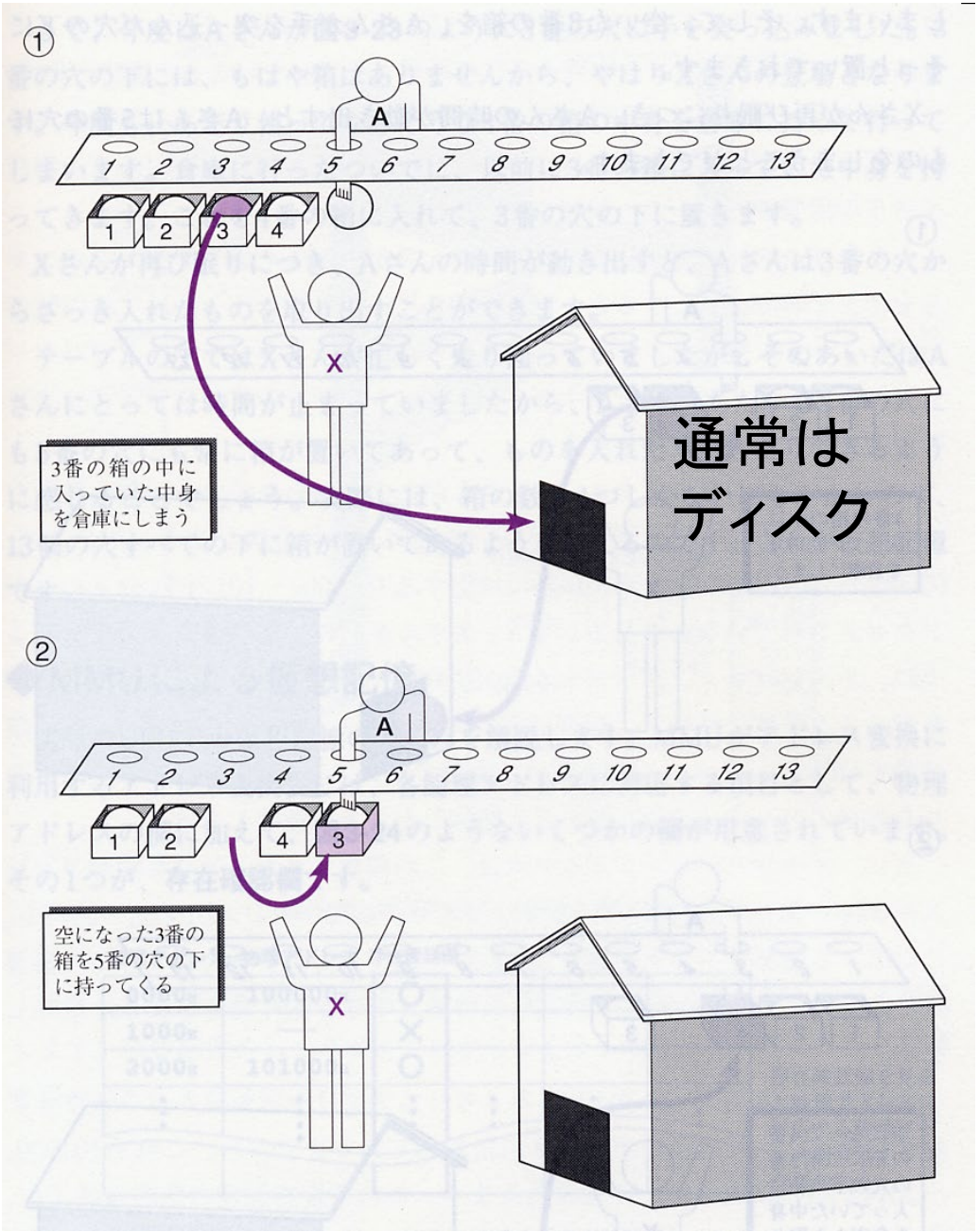
- 18エクサBのメモリ(64bitフルアドレス)を積んだマシンなど無い.
  - いまどきは4GB(32bitフル)くらいはメモリがありますよね.
- そこで，実メモリ(たとえば2GB程度)を使って，4Gのメモリ空間を表現できなければならない.
  - (実際，4Gまで使うかは別として)
- そのための機構として仮想記憶がある.

仮想記憶 1



文献6 p.87

仮想記憶 2



左図のようにメモリ内容が，一時退避されることを，「スワップアウト」(Swap out)と言う.

スワップアウトページの選択

- i386のLinuxではpage単位で行う.
- 最後にアクセスされてから最も長く使われてないページからスワップアウトする.
  - LRU (Least Recent Used)と呼ばれる.
  - LRUを実現するアルゴリズムは多数存在する(そうだ).

スワップインとデマンドページング

- スワップアウトしたページ(にあるデータやコード)も実行途中に再度，必要になるかもしれない.
- この場合，スワップされたページが再度実メモリに配置されることをSwap In と呼ぶ.

- 実はLinuxでは，プログラム実行開始時に全コードとデータを実メモリ上に配置するのではなく,
- 当面必要な部分のみ配置する.
- このような仕組みをデマンドページング(Demand Paging) と呼ぶ.

スラッシング

- 論理記憶に比べ実記憶が極めて小さく，かつ，多数のプロセスがスワップインを必要な場合，
- 当然，仮想記憶によるページのスワップアウトが頻繁に起こる.
- 既に述べたように，ディスク等はメモリよりもかなり遅いため，仮想記憶利用により，処理がかなりもたつく.
- このような現象をスラッシングと呼ぶ.

実際の仮想記憶利用の割合

- 別ページの実データを参照.
  - 4Mページ 64個 4Kページ 2191個
    - 4Mページは4Kページの1024倍 (2^{10}倍)
  - Swap out 4Kが16個だけ
- 実際にはそんなに大きな割合ではない.
- 16/(2191+64*1024)=0.00236%のSwap率
- ちなみに，4Gの全メモリ空間に対して，存在するページの割合は，

  (2191+64*1024)/1048576= 6.4%

授業の感想レビューへ

アンケートのほう，
よろしくご提出ください