オペレーティングシステム
i386アーキテクチャ(3)

2005年10月28日

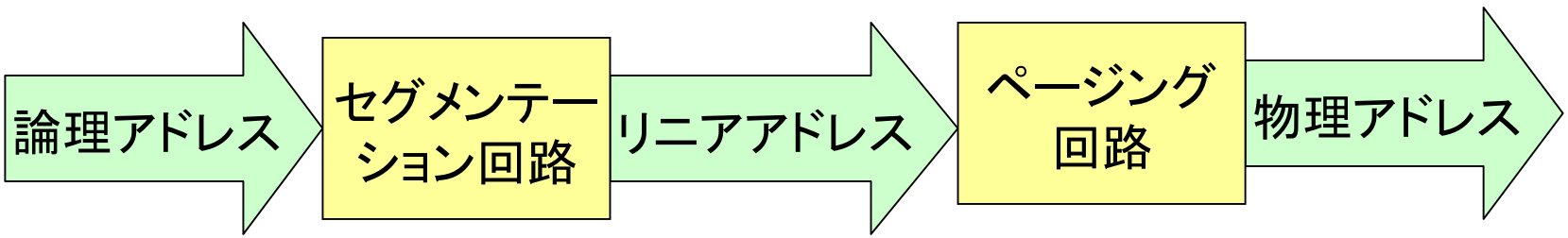
海谷 治彦

目次

- セグメントとページの保護機能
- ページングの詳細
- システムコールが実行される道程
- プロテクトモード以外のモード

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ではそれぞれの段階で保護チェックが行われ，チェックがコケる(許可されてないアクセスがある)と，例外が発生する．

セグメントの保護チェック 1/2

- リミットチェック
  - セグメントはオフセットとサイズがあるので，その範囲の外をアクセスしてないかチェックする.
  - Linuxで使うセグメントでは必ずチェックを通る.
- タイプチェック
  - 各セグメントは実行可能，読み可能，書き可能の型の違いで8種類に分類される.
  - CS, DS等のセグメントレジスタにセグメントを設定する場合，
    - CS 実行可能なセグメントか？ (タイプ 4～7)
    - DS 読み書き等が可能なセグメントか？ (タイプ 0～3)

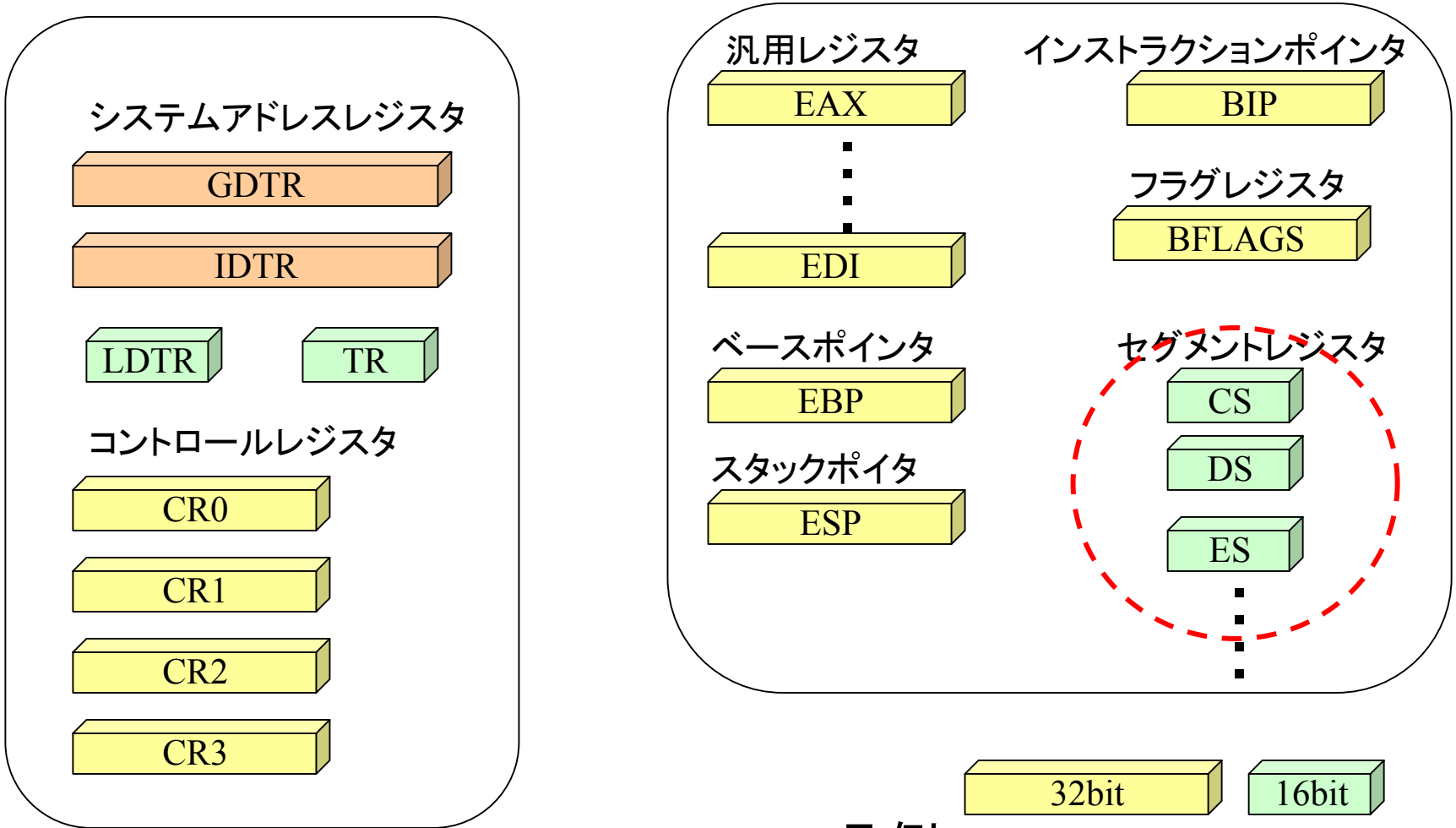
    を判定し，条件にあわないかどうかをチェックする.
    - Linuxの場合，前述の通り，1と5しか使ってない.

セグメントの保護チェック 2/2

- 特権レベルチェック (Privilege Level Check)
  - セグメント毎に設定されたDPLをもとに実行権限をチェックする.
  - 以下の用語に注意
    - CPL (Current PL: 現行特権レベル)
      現時点でCSレジスタが指しているセグメントのDPLの値.
    - DPL (Descriptor PL: デスクリプタ特権レベル)
      セグメントディスクリプタ毎に記述された特権レベル.
      Linuxの場合, KERNEL_*は0, UER_* は3
    - RPL (Requested PL: 要求される特権レベル)
      特権レベルを下げるため. (後述)
      セグメントを指すレジスタ(CS, DS等)の下位2bitにおかれる.
- PL(特権レベル)は0ほど強いことに注意.
  - 3が最弱.

再録

i386のOS関係のレジスタ



DSを更新する際の保護機能

- 以下の場合は違反となる.

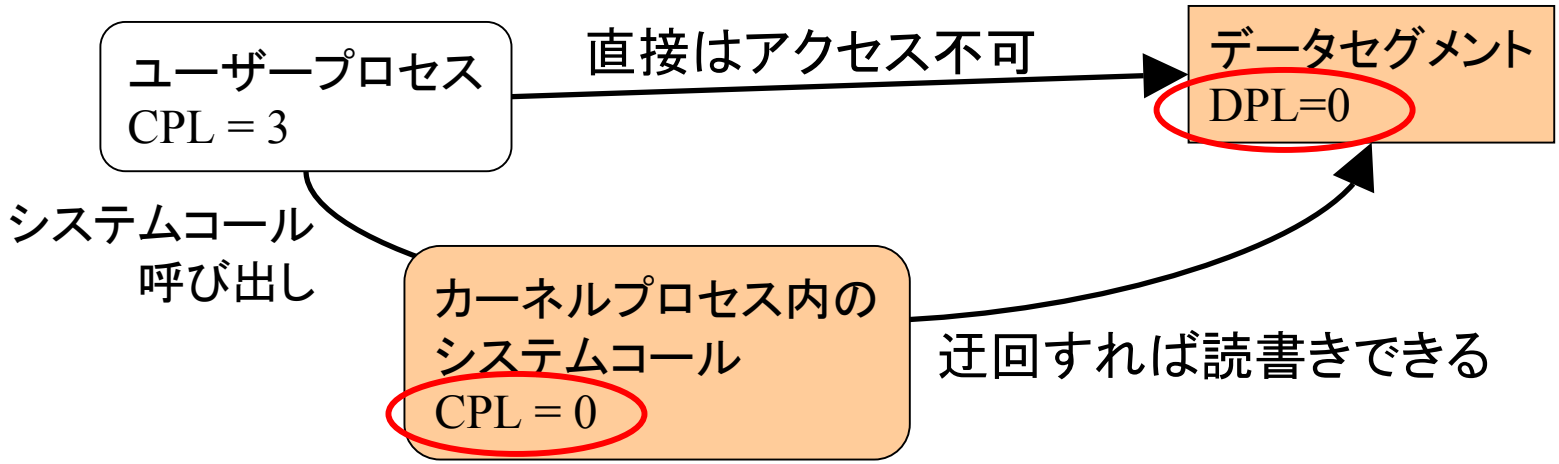
  CPL > DPL

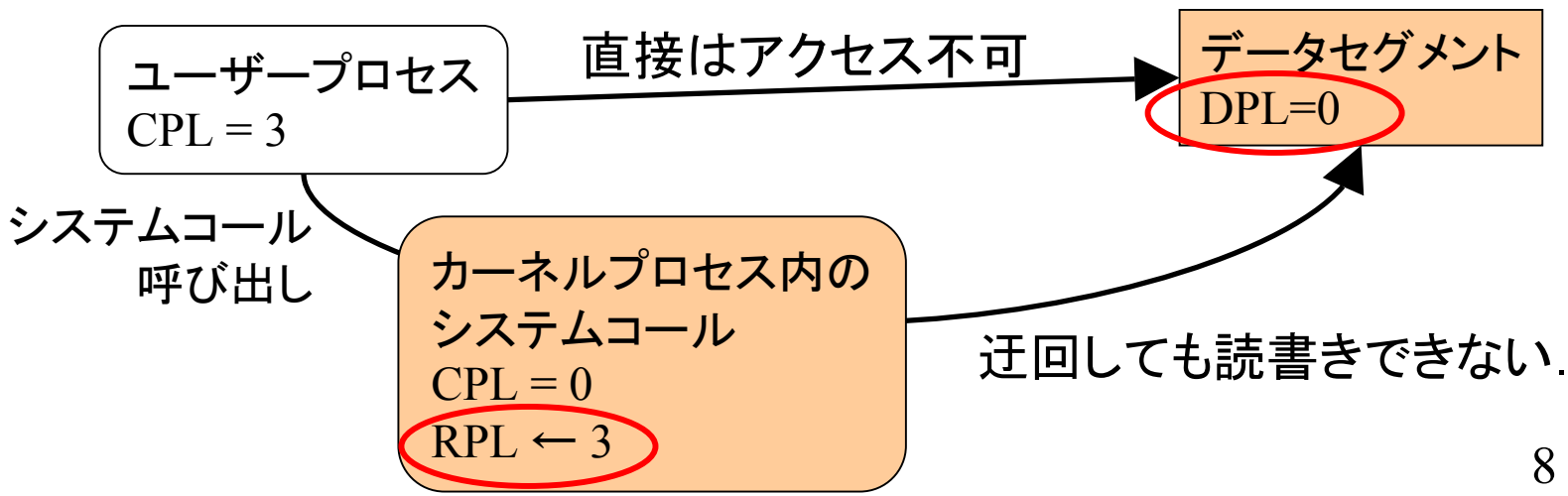
  - 現在のCSの指すセグメントのPLが，扱おうとするセグメントのPLより大きいと違反になる.
  - 例えば，ユーザーレベルのコードが，カーネルのデータを読もうとした場合.

  RPL > DPL

  - システムコールを利用し，ユーザーコードがカーネルコードに実行依頼をすることで，アクセス権限が無いデータにアクセスできてしまうのを防ぐため.
  - 例は後述.

RPLの存在意義の例


ユーザープロセス
CPL = 3
直接はアクセス不可
データセグメント
DPL=0
システムコール
呼び出し
カーネルプロセス内の
システムコール
CPL = 0
迂回すれば読書きできる


ユーザープロセス
CPL = 3
直接はアクセス不可
データセグメント
DPL=0
システムコール
呼び出し
カーネルプロセス内の
システムコール
CPL = 0
RPL ← 3
迂回しても読書きできない.

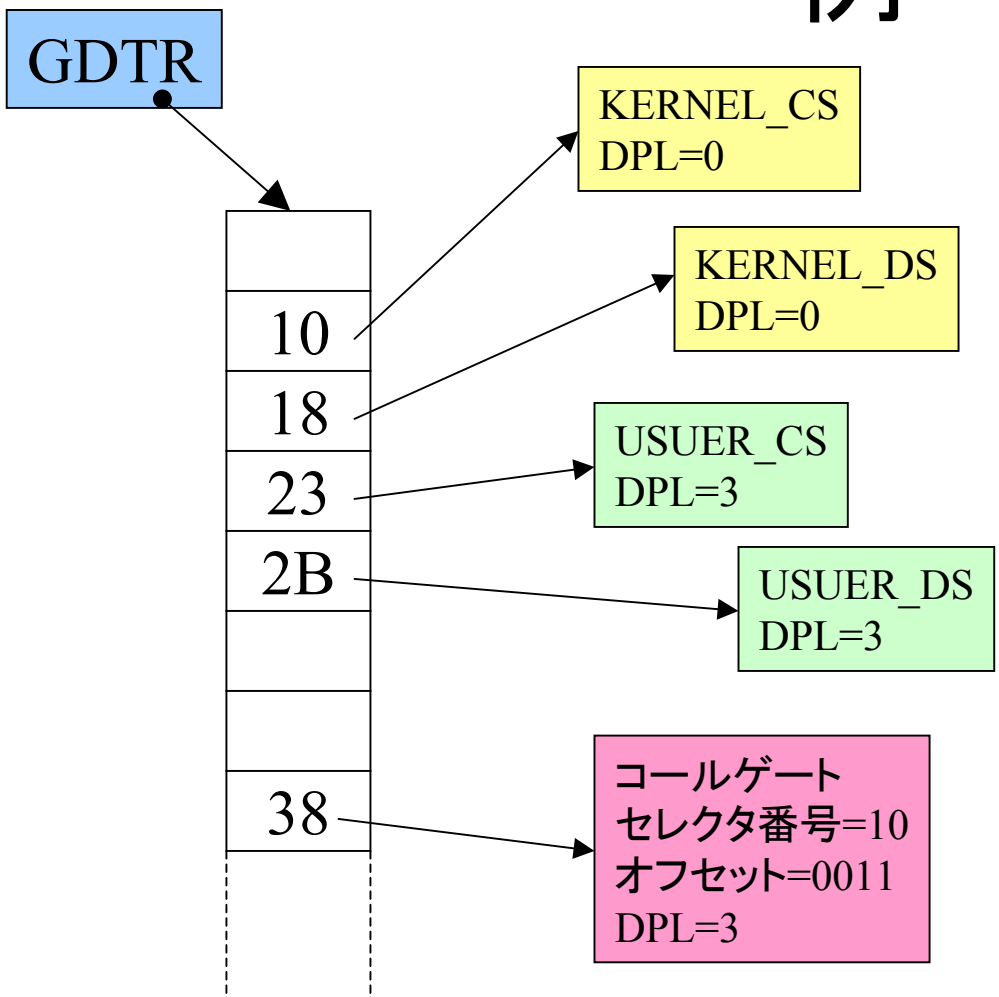
CSを更新する場合の保護機能

- CSレジスタ自体を直接変更する命令は無い.
- CSレジスタはJMP, CALL命令や割込みにより変化する(可能性がある).
- 移行先のDPLがCPLと一致している場合に限り, 移動が許される.
  - それ以外はアクセス違反となる.
- とは言え, このままではユーザープロセス, カーネルプロセスの切り替えが全くできないので, ゲートと呼ばれる機構でCSのPLの移行を可能にする.

コールゲート

- 数種類ある内のゲートの一つ.
- CPLを変更する手段の一つ.
- GDTRが指すDescriptor Tableは，実はSegment Descriptorだけでなく，Gate Descriptor というのも列挙できる.
- 個々のGate Descriptorには，
  - 他のSegment Descriptor のセレクタ値 (番号)
  - そのSegment内での(オフセット)アドレス
  - が記載されており，call 命令 (サブルーチンを呼ぶアセンブラ命令)でGate Descriptorを指定することで，CPLとDPLの一致を無視して他のコードに制御を移行できる.

例



オフセット部分は
捨てられている.

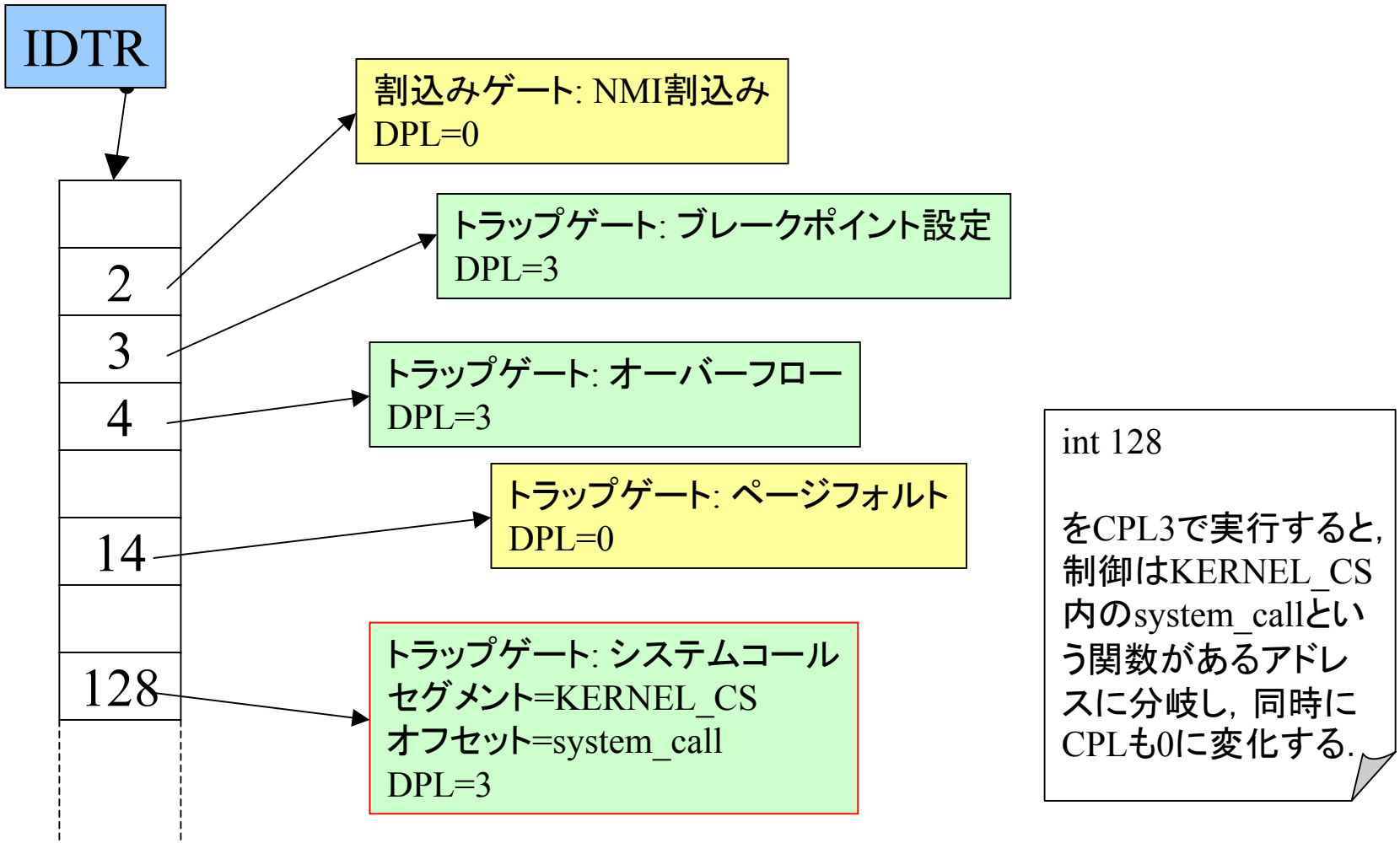
call 0038:00000
命令で,
実際には,
KERNEL_CS内の
オフセット0011番に
あるコードに制御が
移行し,
かつ, CPLも(現CPL
に関係なく)0になる.

※ Linux稼動中にはコールゲートは利用されていないようです. (未確認ですが)

トラップゲート再び

- トラップゲートはLinuxでの例外ハンドラを記述するのに使われていた.
- トラップゲートは，コールゲートと同じ性質を持っている.
  - よって，トラップゲート記述内の飛び先(ハンドラー)をKERNEL_CS内にしておけば，USER_CSから(レベル3)からでも，レベル0の機能を呼び出せる.
  - 例外ハンドリングかサブルーチンコールかの違いだけ.
- (前述のように)システムコールの呼び出しは，この「ゲート」によってi386では実現されている.

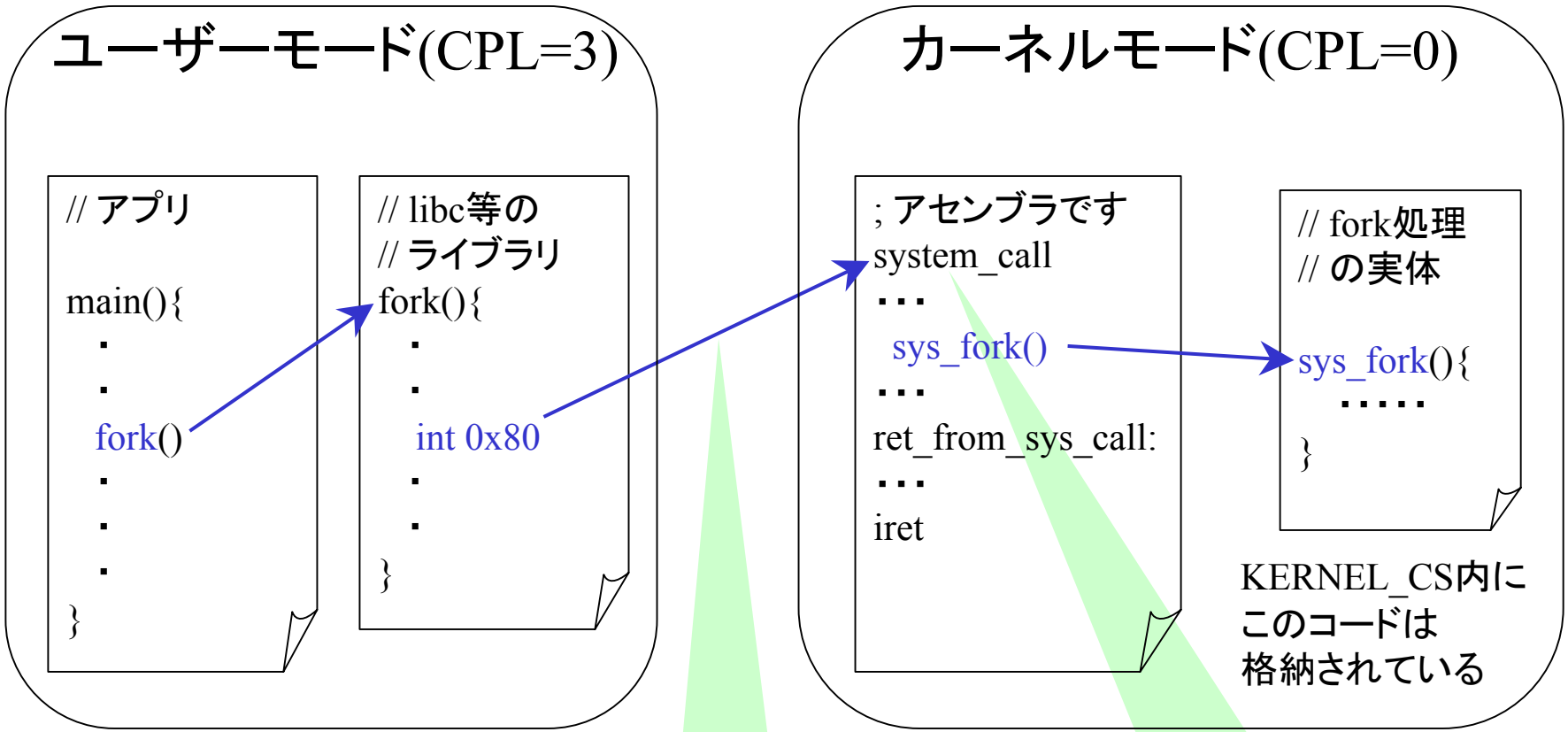
例


IDTR
2
3
4
14
128
割込みゲート: NMI割込み
DPL=0
トラップゲート: ブレークポイント設定
DPL=3
トラップゲート: オーバーフロー
DPL=3
トラップゲート: ページフォルト
DPL=0
トラップゲート: システムコール
セグメント=KERNEL_CS
オフセット=system_call
DPL=3
int 128
をCPL3で実行すると，制御はKERNEL_CS内のsystem_callという関数があるアドレスに分岐し，同時にCPLも0に変化する．

システムコールが呼ばれるまで

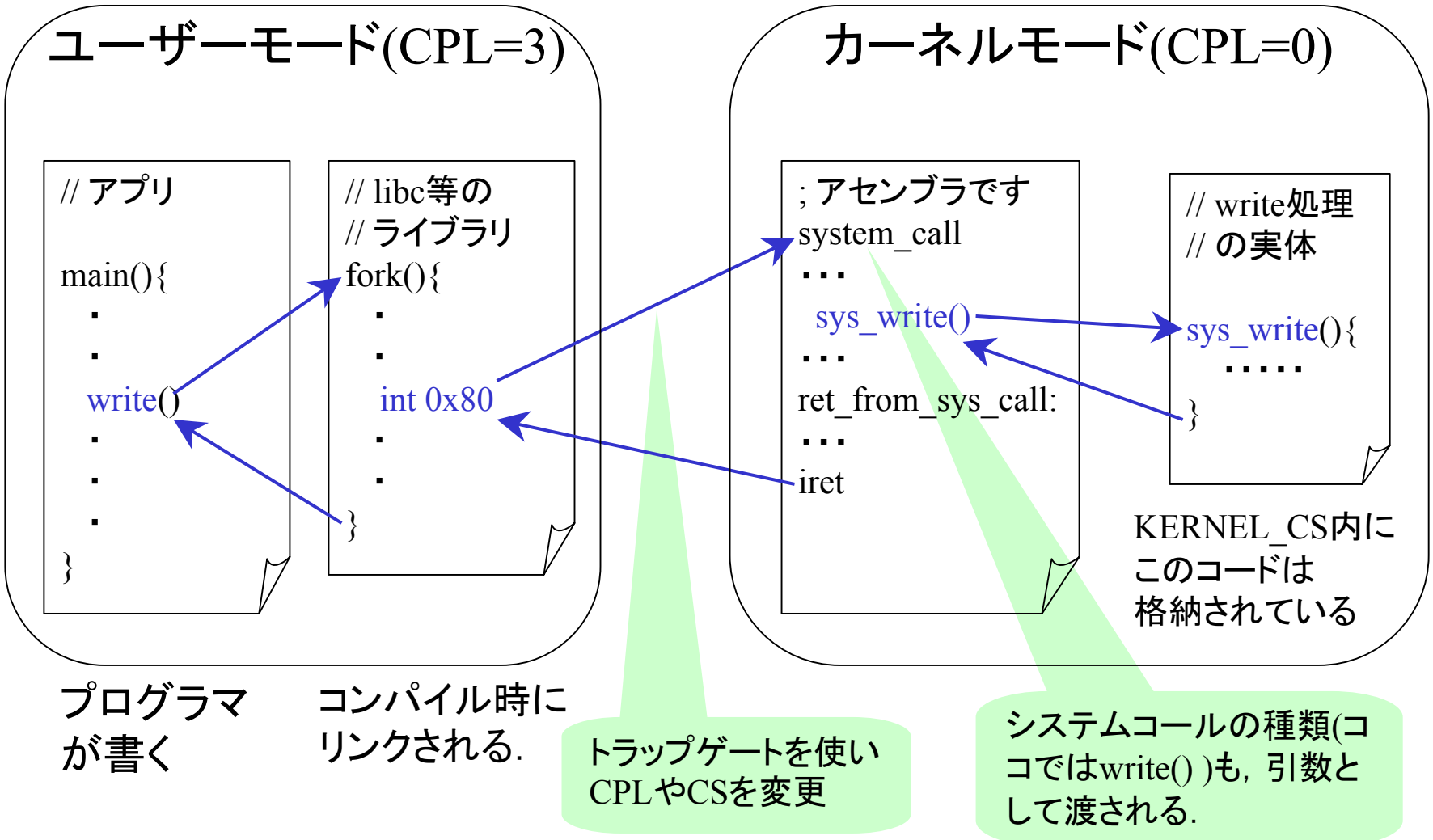
- 前述のようにトラップゲートを利用して，CPLを変化させることで，システムコールは実現されている．
- では，次項で具体的にユーザープログラム内のシステムコールの呼ばれる手順を追う．

例えばfork()



※ fork()は返らないので不適切な例でした.

例えばwrite()


ユーザーモード(CPL=3)
// アプリ
main(){
write()
}
// libc等の
// ライブラリ
fork(){
int 0x80
}
カーネルモード(CPL=0)
; アセンブラです
system_call
sys_write()
ret_from_sys_call:
iret
// write処理
// の実体
sys_write(){
}
KERNEL_CS内に
このコードは
格納されている
プログラマ
が書く
コンパイル時に
リンクされる.
トラップゲートを使い
CPLやCSを変更
システムコールの種類(コ
コではwrite())も，引数と
して渡される.

ライブラリのリンク (参考)

- それぞれのアプリケーションは，いくつかのライブラリがリンクされている.
- リンクされているライブラリは ldd コマンドで確認できる.
- 各ライブラリ内にある関数(というか外部からアクセス可能なシンボル)は，nm コマンドで確認できる.

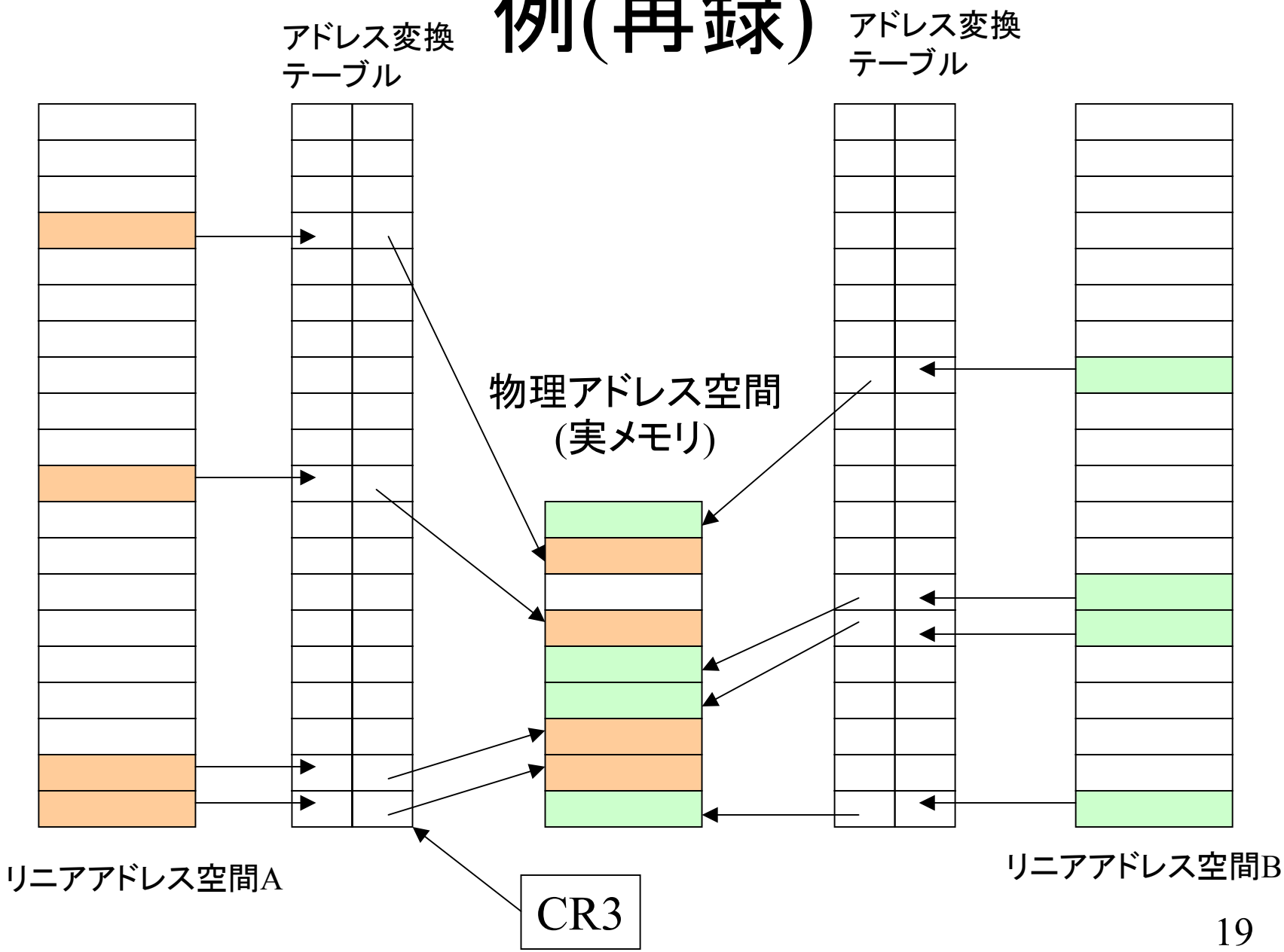
ページのアドレス変換テーブル

- 前述のように，リニアアドレスと実アドレスの変換にテーブルを使っていた.
- しかし，1ページが4KBの場合，

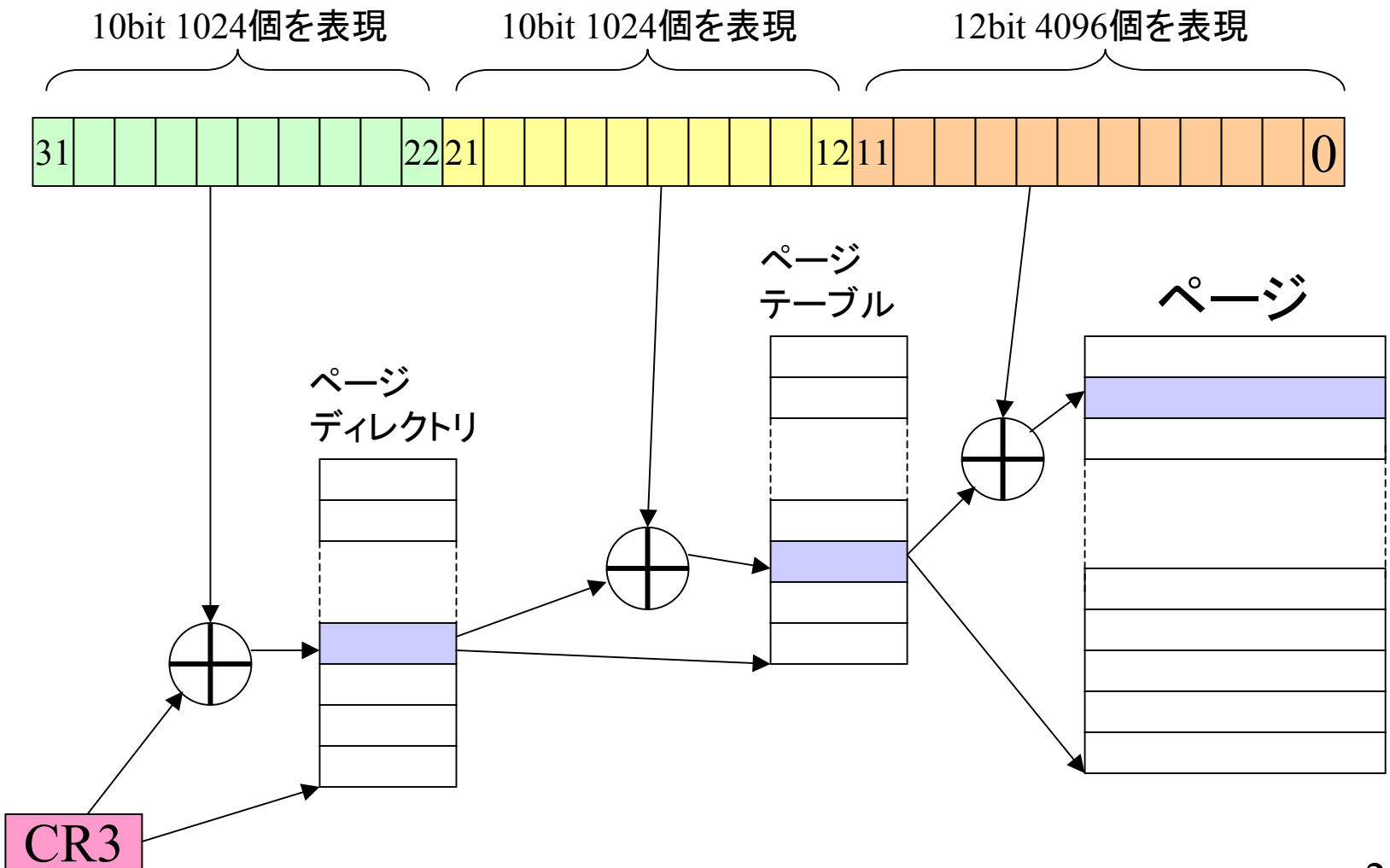
  $4G \div 4K = 2^{32} \div 2^{12} = 2^{20} \fallingdotseq 100$万個

  もの表項目がいる.
- コレは無駄なので，実際には二段階の表となっている.
  - ページディレクトリ 一段目の表
  - ページテーブル 二段目の表
- 実際のページ配置は「すかすか」なので，このほうが効率的.

例(再録)


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

アドレス変換の概念図


10bit 1024個を表現
10bit 1024個を表現
12bit 4096個を表現
31
22
21
12
11
0
ページ
テーブル
ページ
ページ
ディレクトリ
CR3

ページフレームの保護

- 4KBに区切られた個々のページをページフレームと呼ぶ.
- i386のページング機能はオフにできる.
- CPUのCPLが
  - 0～2 の場合，スーパーバイザモード
  - 3の場合，ユーザーモードと呼ぶ.
- ページフレームにはU/SフラグとR/Wフラグの二つがあり，これらの値をCPUのモードによってアクセス権限がかわる.

U/SフラグとR/Wフラグ

U/Sはアクセス可能性を制限

CPUのモード	U/S=0	U/S=1
ユーザー	アクセス不可	アクセス可
スーパーバイザー	アクセス可	アクセス可

R/Wはアクセス可能性を制限

CPUのモード	R/W=0	R/W=1
ユーザー	読み出し専用	読書き可能
スーパーバイザー	読書き可能	読書き可能

プロテクトモード以外のモード

- リアルモード
  - i386のマシンは電源投入時にはこのモードで動く.
  - アドレス空間が狭い. 1MB (20bit)
  - 保護機能は無い.
  - LinuxではOS起動準備のみに使う.
- 仮想8086モード
  - プロテクトモードでリアルモードのプログラムを動かすためのモード.
  - DOSのプログラムがWinのコマンドプロンプトで動作しちゃうのはこのせい.