オペレーティングシステム

第9回

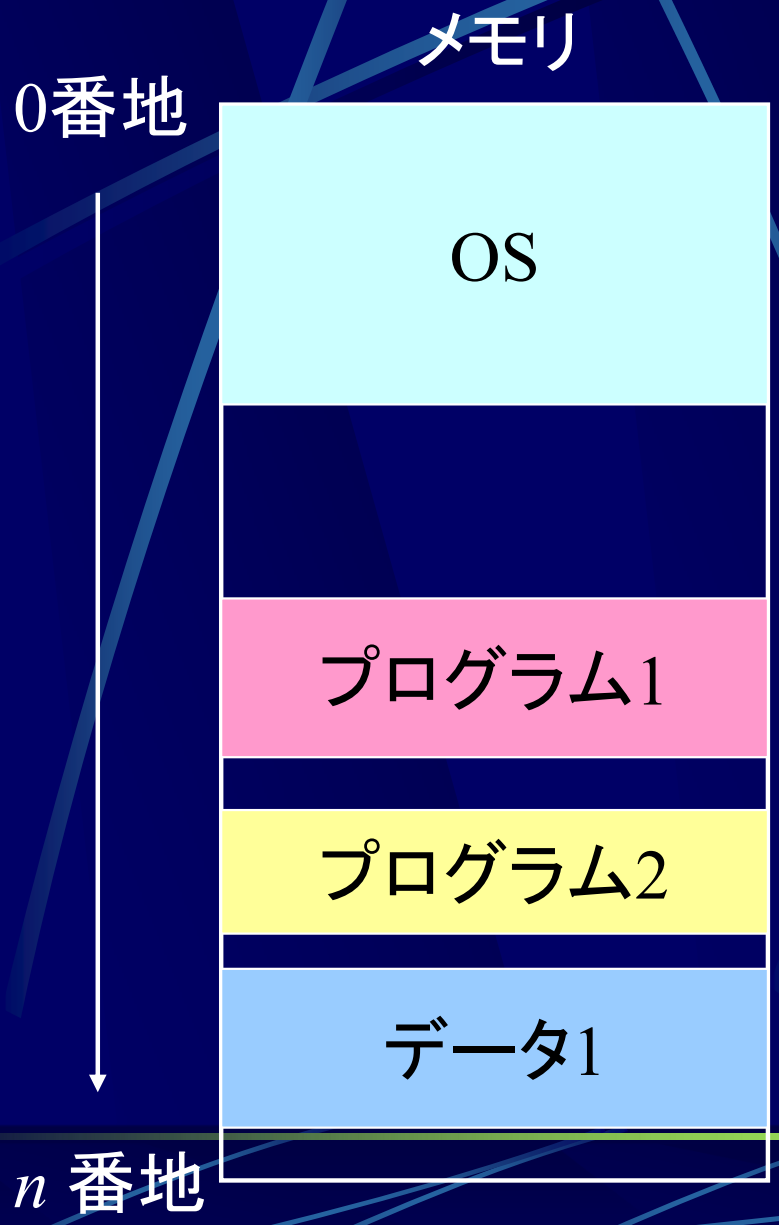
仮想記憶管理(1)

http://www.info.kindai.ac.jp/OS

E号館3階E-331 内線5459

takasi-i@info.kindai.ac.jp

メモリ

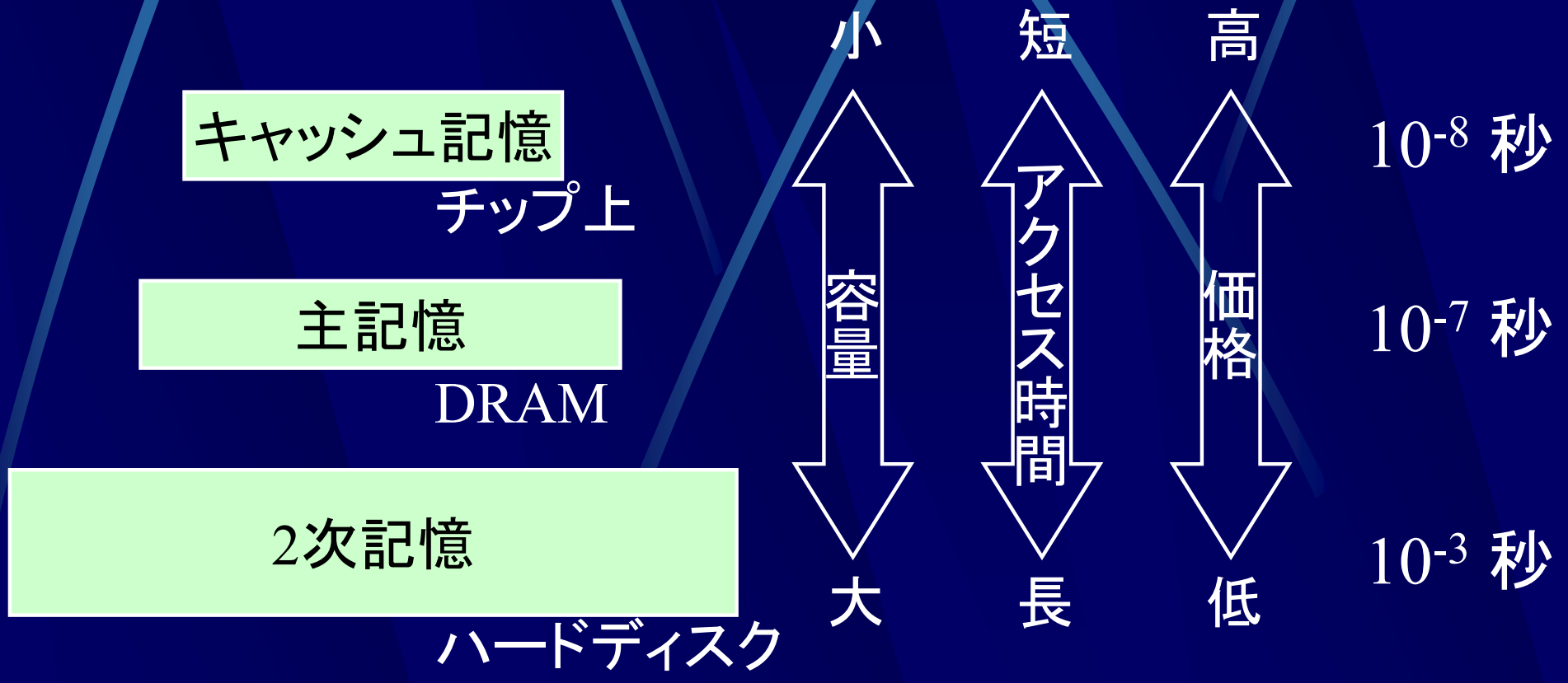


OS, ユーザプログラム, データ
メモリ上に置かれる

メモリ上の位置は
1 次元のアドレスで管理

メモリ

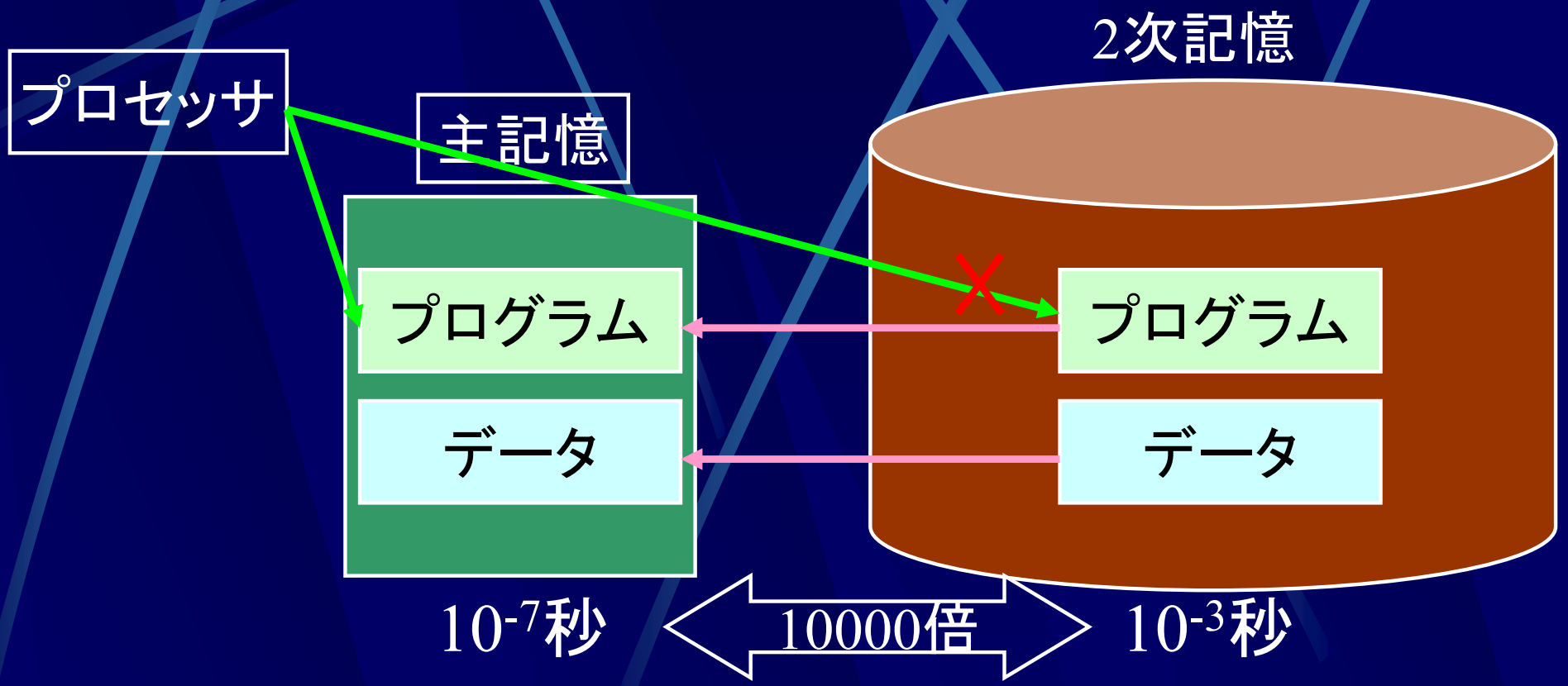
- メモリの記憶階層



メモリ

記憶装置	本, 資料	特徴
キャッシュ (cache memory)	手で保持 	すぐ読める ごくわずかな量しか持てない
主記憶 (main memory)	作業机 	座ったまますぐ手に取れる 置ける量は限られる
2次記憶 (secondary memory)	倉庫 	部屋を出て取りにいく必要あり 大量に置ける

主記憶と2次記憶

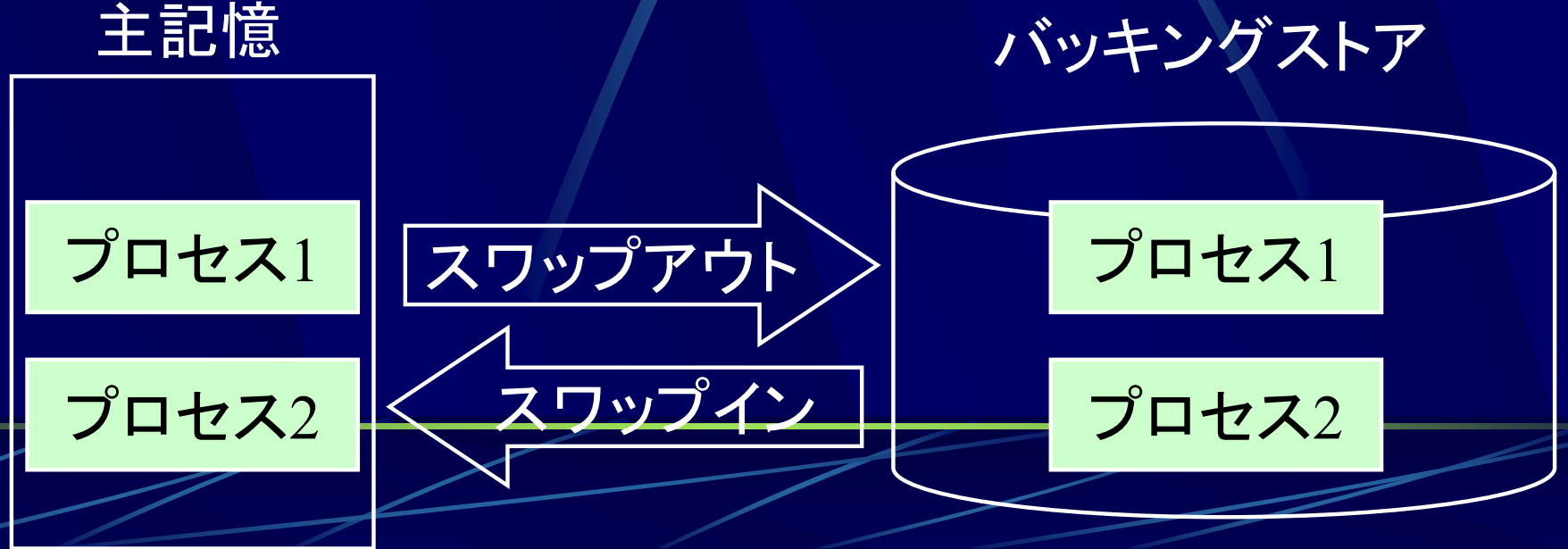


プロセッサは2次記憶を直接読むことはできない

使用するプログラム，データは主記憶上にコピー

スワッピング(swapping)

- スワッピング(swapping)
  - 待ち状態のプロセスを2次記憶に退避させる
- バッキングストア(backing store)
  - スワッピングを行う際に使用する2次記憶

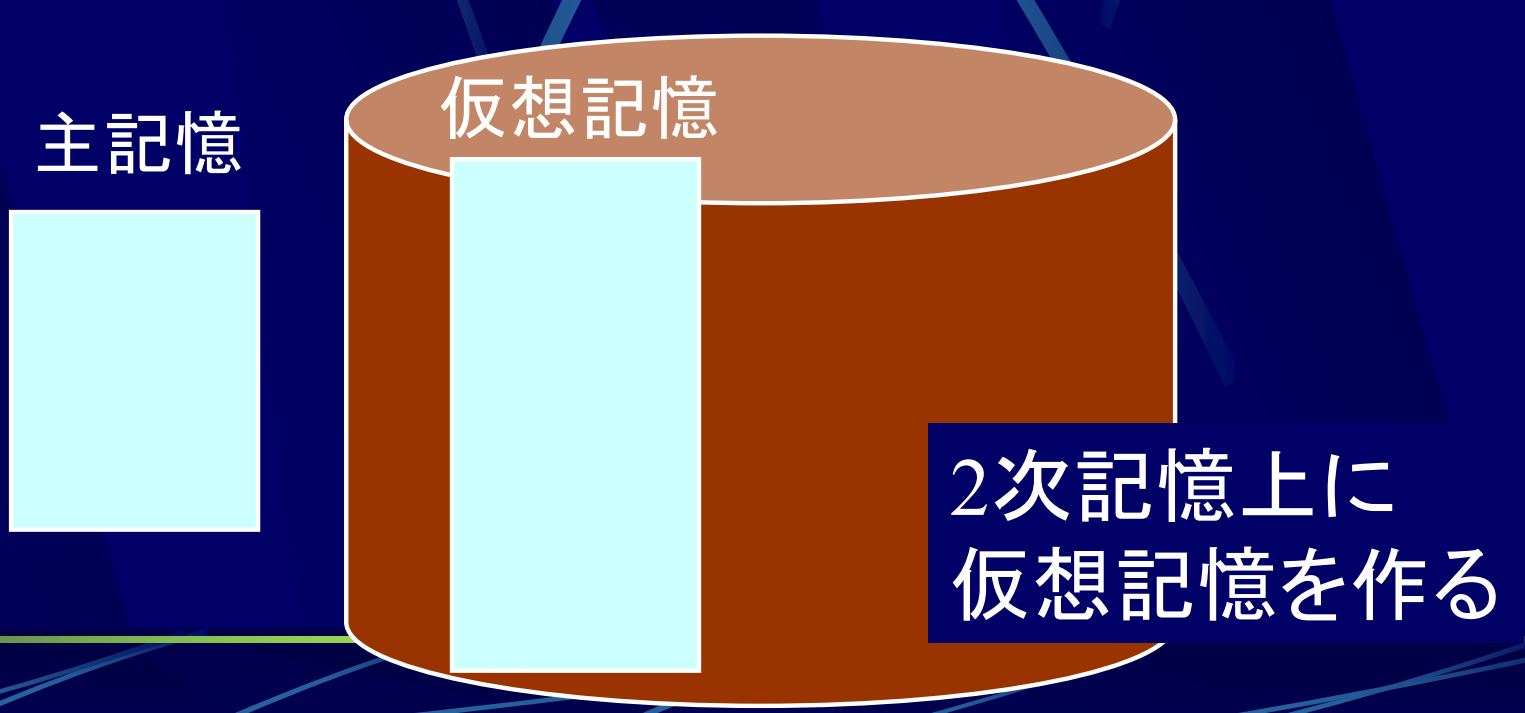


スワップイン, スワップアウト
(swap-in, swap-out)

- スワップイン(swap-in)
  - プログラム, データを2次記憶から主記憶に
  - 実行中のプロセスに必要なものを読み込む
- スワップアウト(swap-out)
  - プログラム, データを主記憶から2次記憶に
  - スワップインの領域を確保するために当面必要の無いものを退避させる

仮想記憶(virtual memory)

- 仮想記憶(virtual memory)
  - 動的再配置により主記憶容量よりも大きなアドレス空間を提供



仮想アドレス(virtual address)

- 仮想アドレス(virtual address)
  論理アドレス(logical address)
  - 仮想記憶で用いられるアドレス
- 実アドレス(real address)
  物理アドレス(phisical address)
  - 実際の主記憶で用いられるアドレス

記憶領域の仮想化

- 記憶領域の仮想化
  - 仮想アドレス空間の一部が実メモリ上に存在

    ⇒ 仮想アドレスと実アドレスの対応付けが必要
  - 実メモリ上に存在する“仮想記憶の一部”は時々刻々と変化

    ⇒ 対応付けも時々刻々と変化

アドレスの動的再配置

アドレス変換

- 動的アドレス変換法
  - ベースレジスタ(base register)
  - ページング(paging)
  - セグメンテーション(segmentation)

アドレス変換表

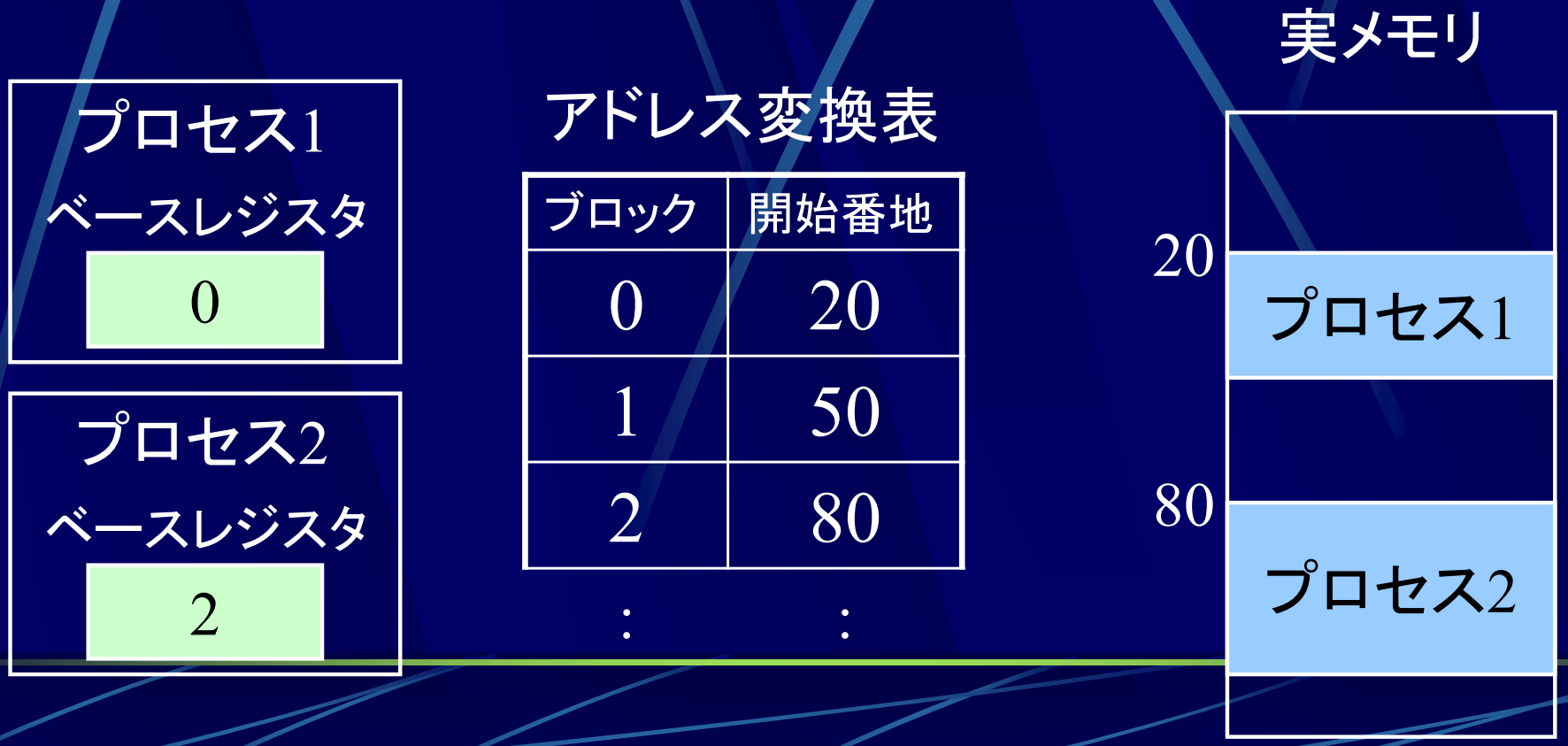
- アドレス変換表
  - 仮想アドレス ⇒ 実アドレスの変換表

バイト単位で表を作ると表が巨大に
(サイズ 1GB の仮想記憶で
変換表を作ると表のサイズが10億)

ブロック単位で表を作る

アドレス変換
ベースレジスタ(base register)

- ベースレジスタ(base register)
  - 起点となるブロック番号を格納



アドレス変換
ベースレジスタ

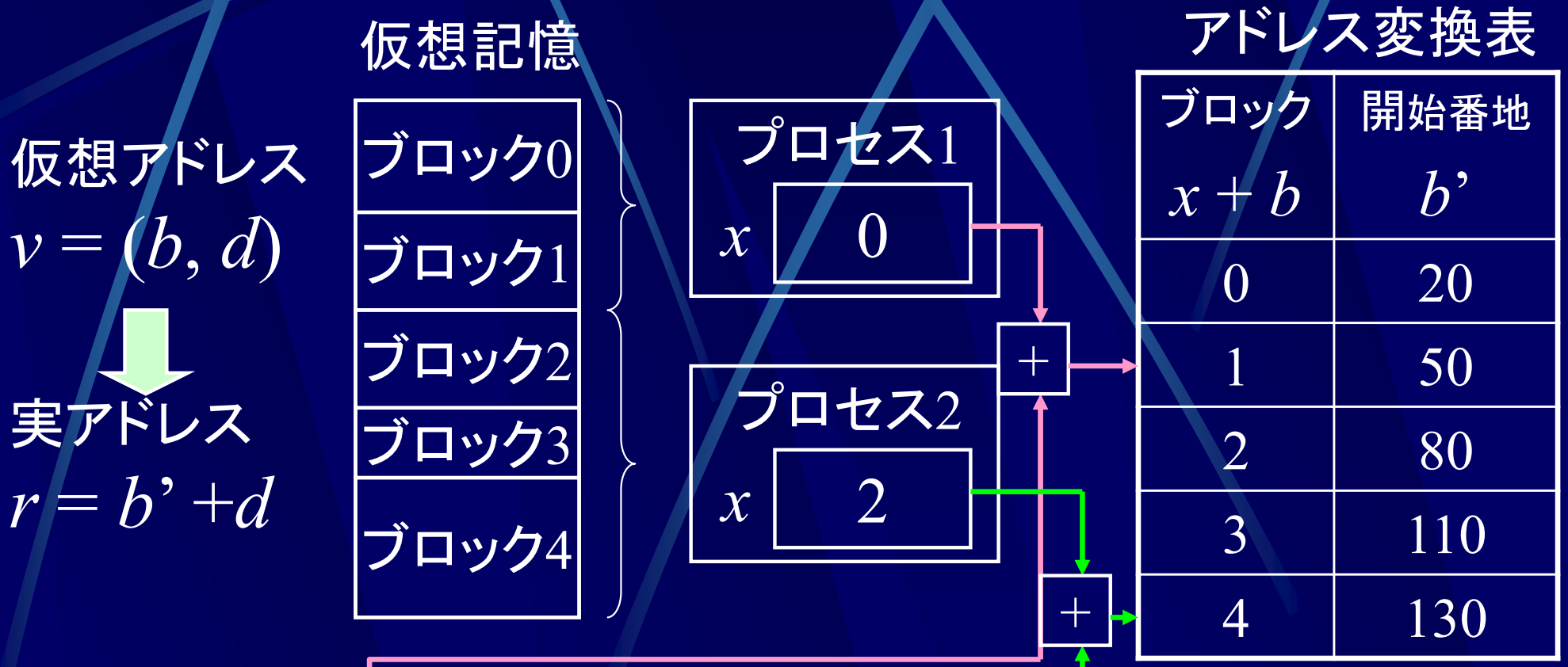
- 仮想アドレス $v = (b, d)$ ⇒ 実アドレス r
  - b : 仮想アドレス上のブロック番号
  - d : ブロックの先頭からの相対位置
  - x : ベースレジスタの値

プロセス1
ベースレジスタ
x

ブロック	開始番地
x	
:	:
$x + b$	b'
:	:

実アドレス
$r = b' + d$

アドレス変換
ベースレジスタ



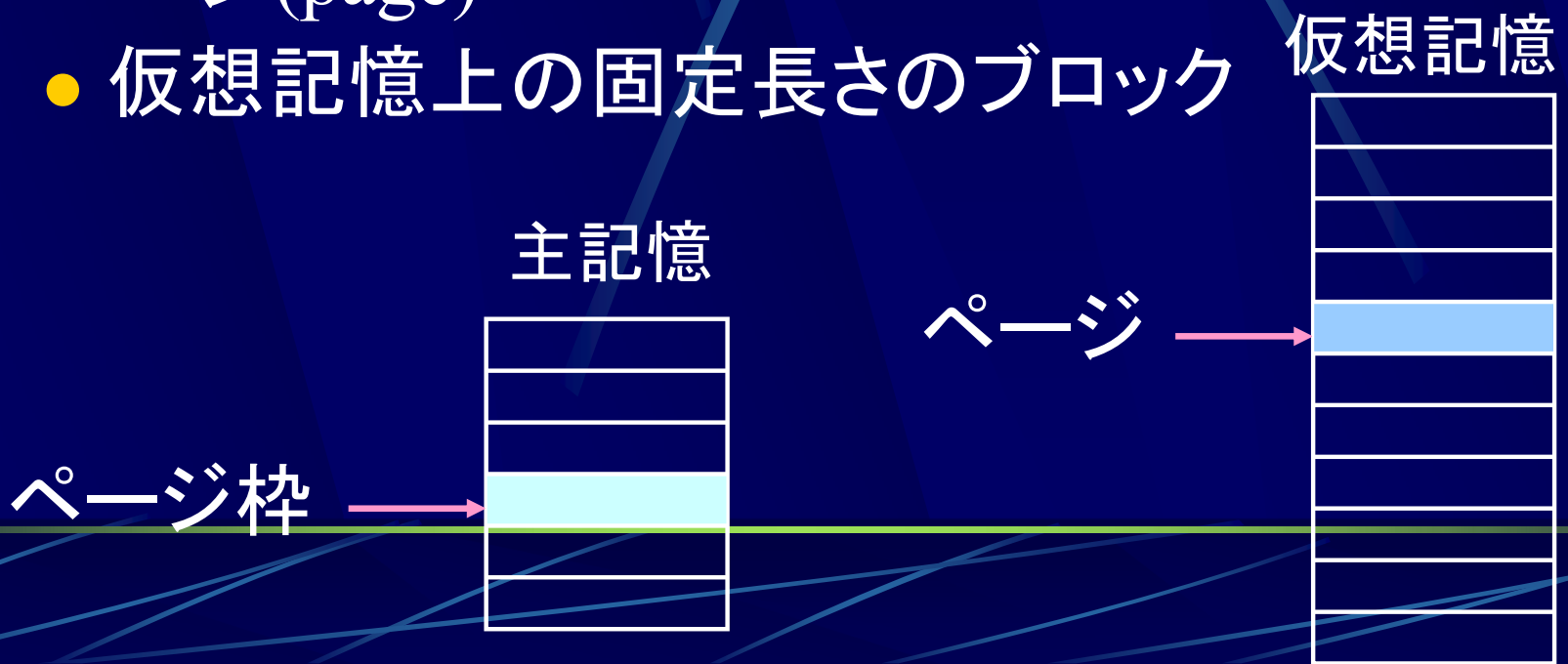
プロセス1の(1, 5)番地　$x = 0, b = 1, d = 5$　$b' = 50$　$r = 55$

プロセス2の(2, 10)番地　$x = 2, b = 2, d = 10$　$b' = 130$　$r = 140$

ページング(paging)

- ページング(paging)
  - 主記憶, 仮想記憶を固定長のブロックに分割
- ページ枠(page frame)
  - 主記憶上の固定長のブロック
- ページ(page)
  - 仮想記憶上の固定長さのブロック

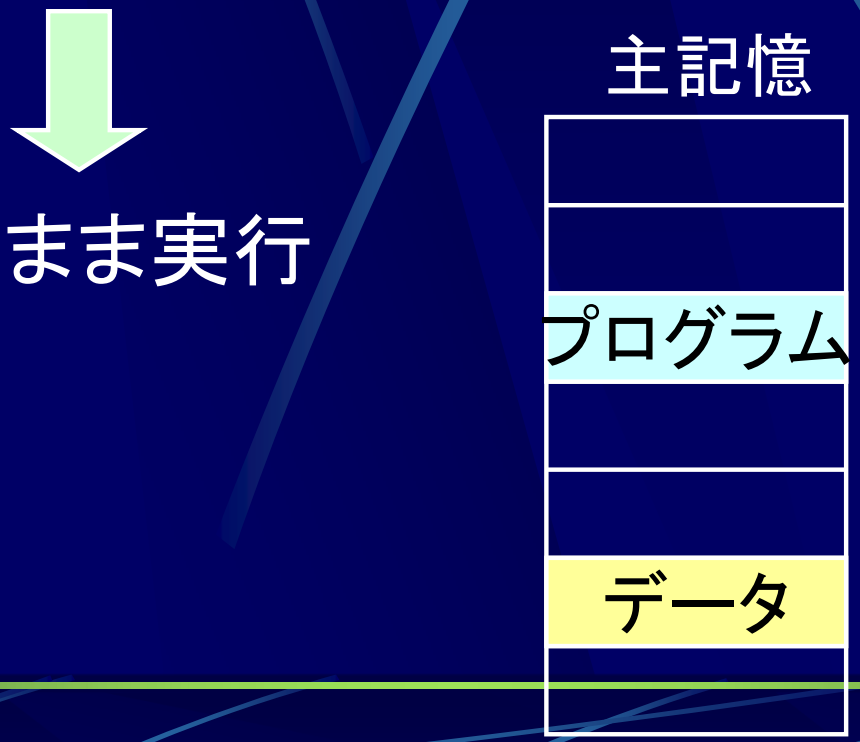
サイズ 2^k KB
4～8KB



ページング

必要なプログラム, データの載っているページが

1. 主記憶上にある場合



そのまま実行

ページング

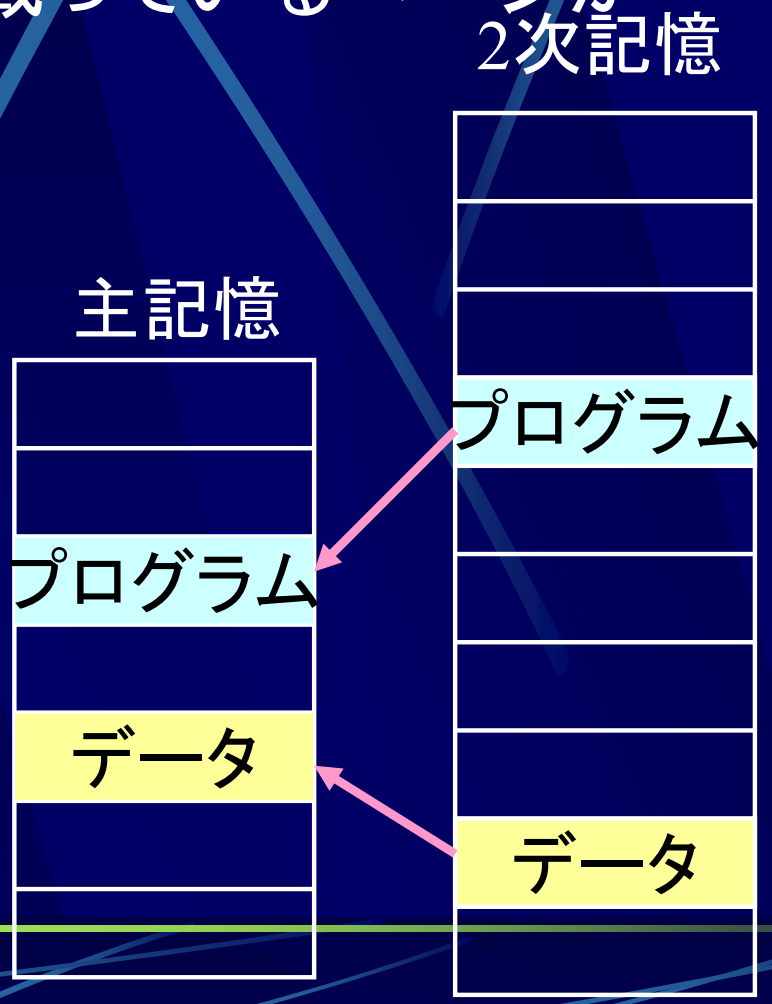
必要なプログラム, データの載っているページが

2. 主記憶上に無い場合

ページフォルト
(page fault)

2次記憶から主記憶に
ページを読み込む

ページイン(page in)



ページング

- ページフォルト(page fault)
  - 必要なページが主記憶上に無い
- ページイン(page in)
  - 2次記憶から主記憶のページ枠にページを読み込む
- ページアウト(page out)
  - ページインの際、使用しないページを主記憶から2次記憶に書き出してページ枠を空ける

ページング

主記憶

ページ枠	実アドレス
0	0000～0FFF
1	1000～1FFF
2	2000～2FFF
3	3000～3FFF

2次記憶

ページ	仮想アドレス
00	00000～00FFF
01	01000～01FFF
02	02000～02FFF
03	03000～03FFF
04	04000～04FFF
0	
0	
0	

ページ	ページ枠
03	2
01	3

仮想アドレス 03357 = 実アドレス 2357

仮想アドレス 01864 = 実アドレス 3864

下位ビットは共通

上位ビットは変換が必要

注意

- ページサイズは4KB (=4096B)か8KB (=8192B)
- 計算機上での処理は2進数

ただし、以降は以下を仮定

- ページサイズは1000B
- 数値は全て10進数

課題テスト・オンライン試験も同じ

ページテーブル(page table)

- ページテーブル(page table)
  - 仮想アドレスから実アドレスへの変換表

ページ (2次記憶)	ページ枠 (主記憶)	フラグ		
		V	P(r,w,x)	C
00		0	1,0,0	0
01	1	1	1,1,0	0
02		0	1,0,1	0
03	2	1	1,1,1	1

ページテーブル

- Vフラグ(virtual memory flag)
  - ページが主記憶上に存在するか否か
- Pフラグ(permission flag)
  - 読み込み可, アクセス可等のアクセス条件
- Cフラグ(change flag)
  - ページイン後、書き込みが行われたか否か

V = 0 なら2次記憶からの読み込みが必要
C = 1 なら2次記憶への書き出しが必要

ページテーブル

ページテーブル

ページ	ページ枠	フラグ		
		V	P	C
00		0	100	0
01	0	1	110	0
02		0	110	0
03		0	110	0
04		0	110	0
05		0	110	0
06	3	1	100	1
07		0	111	0

主記憶

ページ枠	ページ
0	01
1	
2	
3	06

2次記憶

ページ
00
01
02
03
04
05
06
07

ページテーブルの
エントリ(行)数は
仮想記憶の
ページ数と同じ

ページングの動作

主記憶上にある場合

仮想アドレス

01	246

実アドレス

0	246

主記憶

ページ枠	ページ
0	01
1	
2	
3	06

ページ	ページ枠	フラグ		
		V	P	C
00		0	100	0
01	0	1	110	1

主記憶上の位置

主記憶上に有り

2次記憶

ページ
00
01
02
03
04
05
06
07

そのまま主記憶にアクセス可能

ページングの動作

主記憶上に無い場合

仮想アドレス

02	123

主記憶

ページ枠	ページ
0	01
1	
2	
3	06

ページ	ページ枠	フラグ		
		V	P	C
		0	100	0
01		1	110	1
02		0	110	0
03		0	111	0

主記憶上に無し

2次記憶

ページ
00
01
02
03
04
05
06
07

ページフォルト発生

ページングの動作

主記憶上に無い場合

仮想アドレス

02	123

実アドレス

1	123

主記憶

ページ枠	ページ
0	01
1	02
2	
3	06

2次記憶

ページ
00
01
02
03
04
05
06
07

2次記憶から読み込み

ページ	ページ枠	フラグ V	フラグ P	フラグ C
02	1	1	110	0
03		0	111	0

主記憶上の位置

主記憶上に有り

ページングの動作

主記憶上に無い場合

仮想アドレス

00	789

実アドレス

2	789

主記憶

ページ枠	ページ
0	01
1	02
2	00
3	06

2次記憶

ページ
00
01
02
03
04
05
06
07

ページ	ページ枠	フラグ		
		V	P	C
00	2	1	100	0
01	0	1	110	1
02	1	1	110	0
03		0	111	0

ページングの動作

主記憶上に無い場合

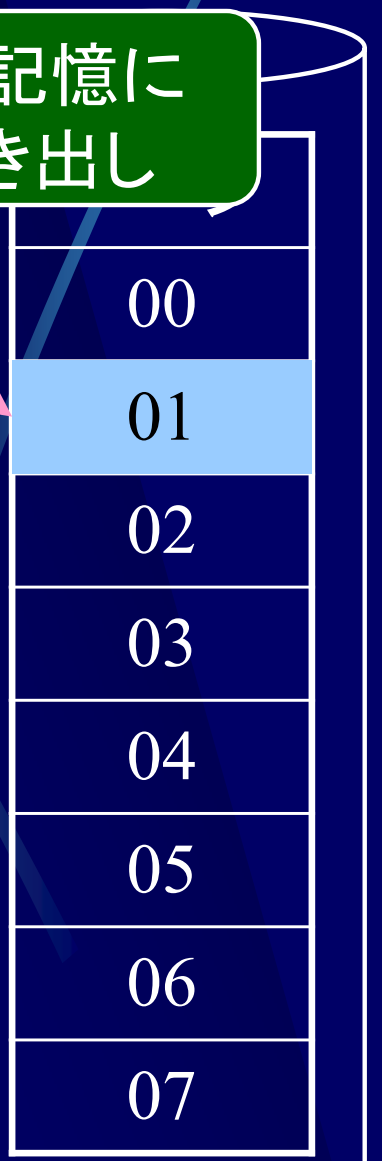
仮想アドレス

03	999

主記憶

ページ枠	ページ
0	
1	02
2	00
3	06

2次記憶



2次記憶に
書き出し

ページ	ページ枠	フラグ		
		V	P	C
00	2	1	100	0
01		0	110	0
02	1	1	110	0
03		0	111	0

ページ枠に
空きが無い
ページアウト

ページイン後
書き込み有り

ページングの動作

主記憶上に無い場合

仮想アドレス

03	999

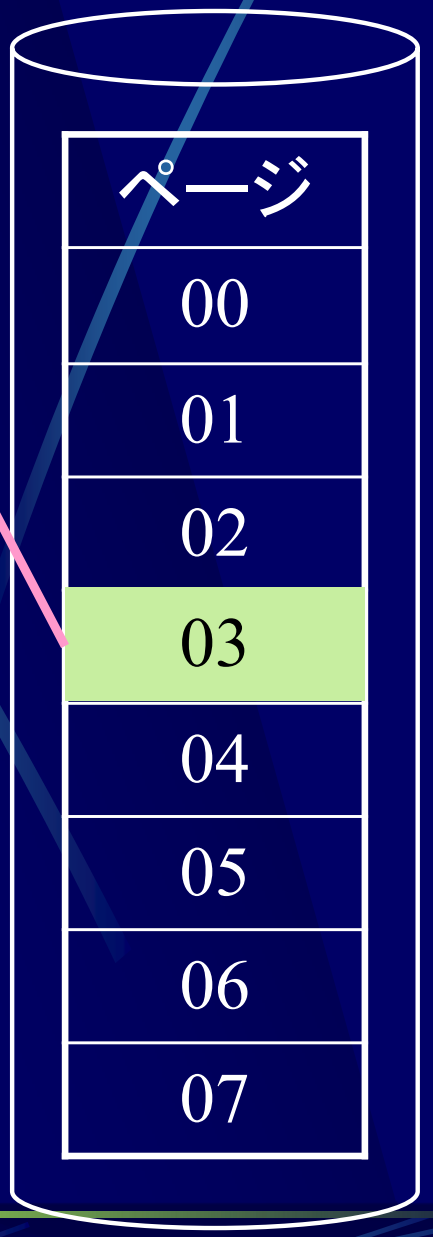
実アドレス

0	999

主記憶

ページ枠	ページ
0	03
1	02
2	00
3	06

ページ	ページ枠	フラグ		
		V	P	C
00	2	1	100	0
01		0	110	0
02	1	1	110	0
03	0	1	111	0



ページングの動作

主記憶上に無い場合

仮想アドレス

01	500

主記憶

ページ枠	ページ
0	03
1	
2	00
3	06

ページ	ページ枠	フラグ		
		V	P	C
00	2	1	100	0
01		0	110	0
02		0	110	0
03	0	0	111	0

2次記憶への
書き出しは不要

ページイン後
書き込み無し

2次記憶

ページ
00
01
02
03
04
05
06
07

ページングの動作

主記憶上に無い場合

仮想アドレス

01	500

実アドレス

1	500

主記憶

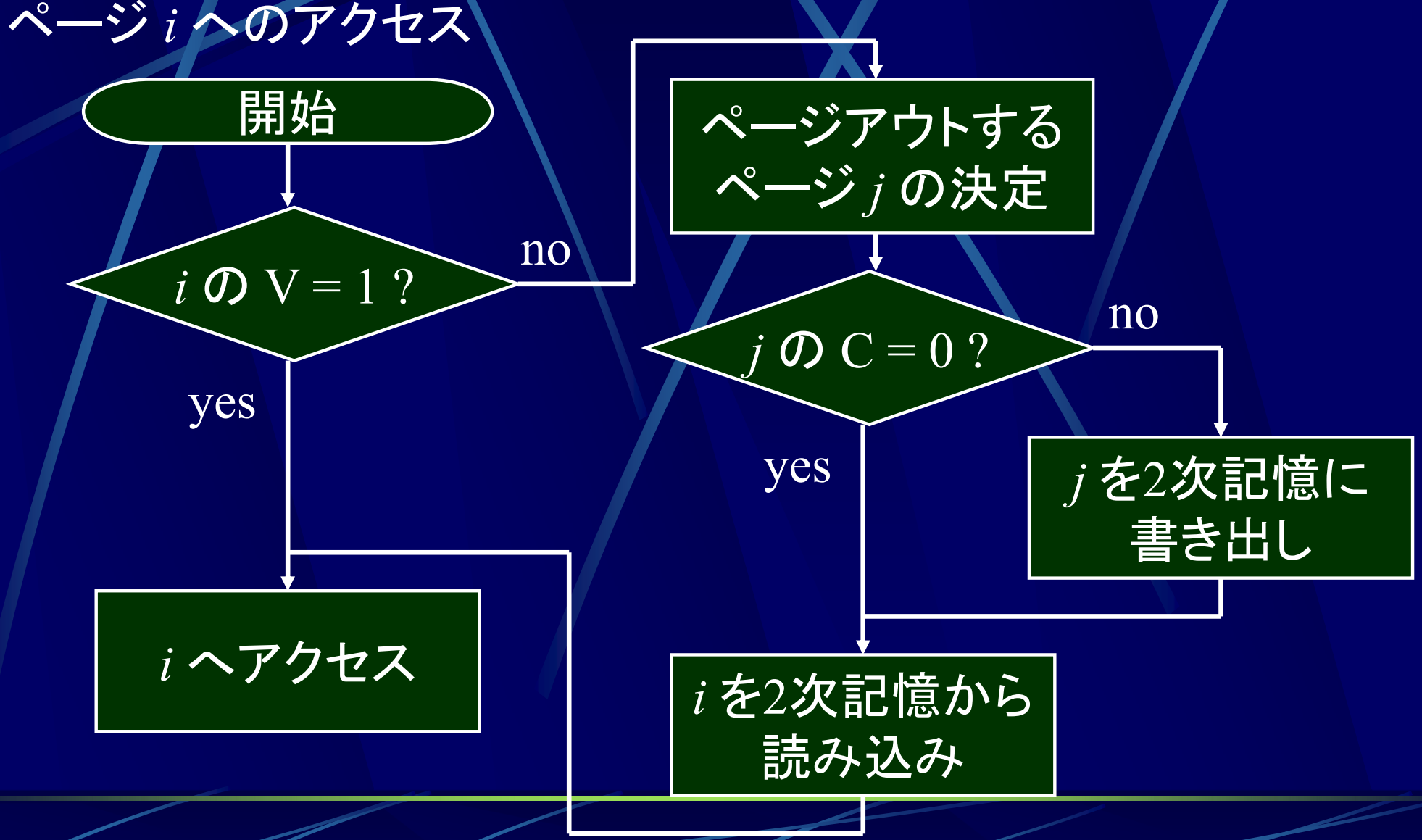
ページ枠	ページ
0	03
1	01
2	00
3	06

2次記憶

ページ
00
01
02
03
04
05
06
07

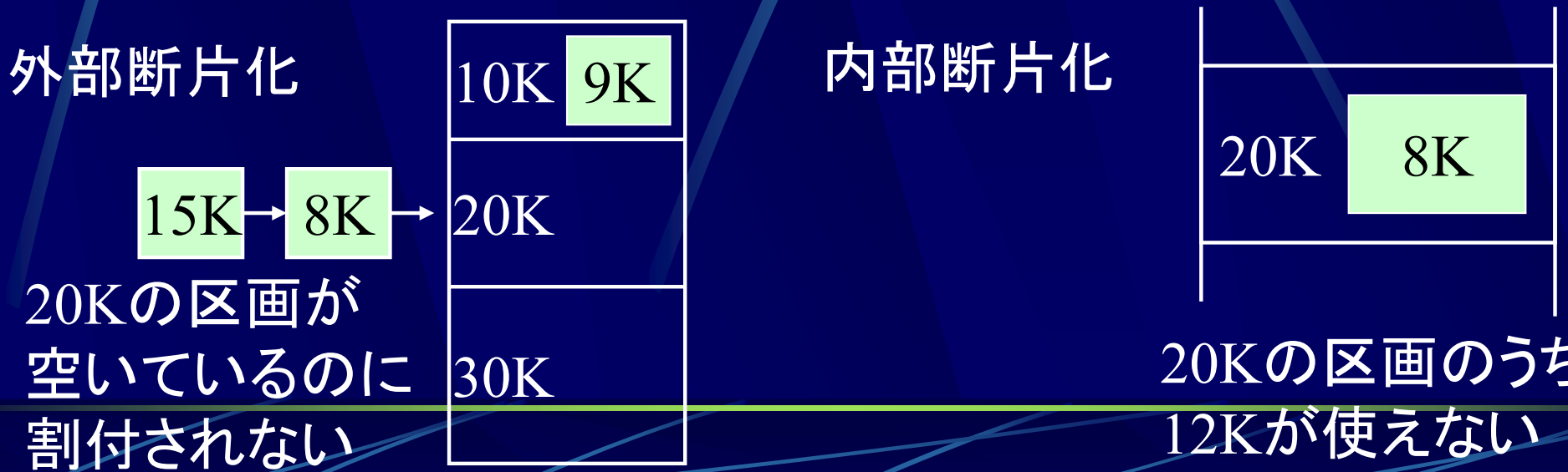
ページ	ページ枠	フラグ		
		V	P	C
00	2	1	100	0
01	1	1	110	0
02		0	110	0
03	0	0	111	0

ページングの動作


ページ i へのアクセス
開始
i の V = 1 ?
no
yes
i へアクセス
ページアウトする
ページ j の決定
j の C = 0 ?
no
yes
j を2次記憶に
書き出し
i を2次記憶から
読み込み

外部断片化, 内部断片化
(external fragmentation, internal fragmentation)

- 外部断片化(external fragmentation)
  - 区画が未使用であるのに割り付けされない
- 内部断片化(internal fragmentation)
  - 区画内で未使用領域がある



ページングの断片化

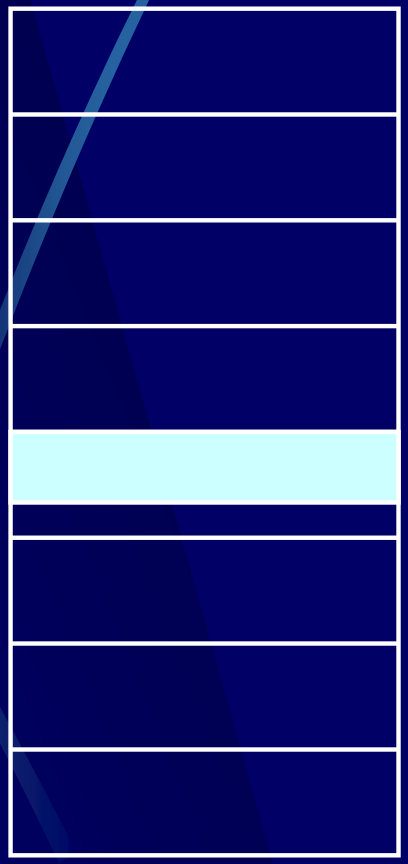
- 外部断片化
  - ページ枠・ページは全て同じサイズ

  ⇒どのページ枠にも入れられる

  ⇒外部断片化は起きない
- 内部断片化
  - ページ枠・ページは全て同じサイズ

  ⇒ページサイズより小さいデータもある

  ⇒内部断片化は起きる



ページングの断片化

- 主記憶はページ単位で割り当てられる

無駄になる領域はごくわずか

ページングの問題点

- ページテーブルが巨大

  例: 仮想記憶を 4GB, 1ページ 8KB で構成

  ページエントリ数 約50万

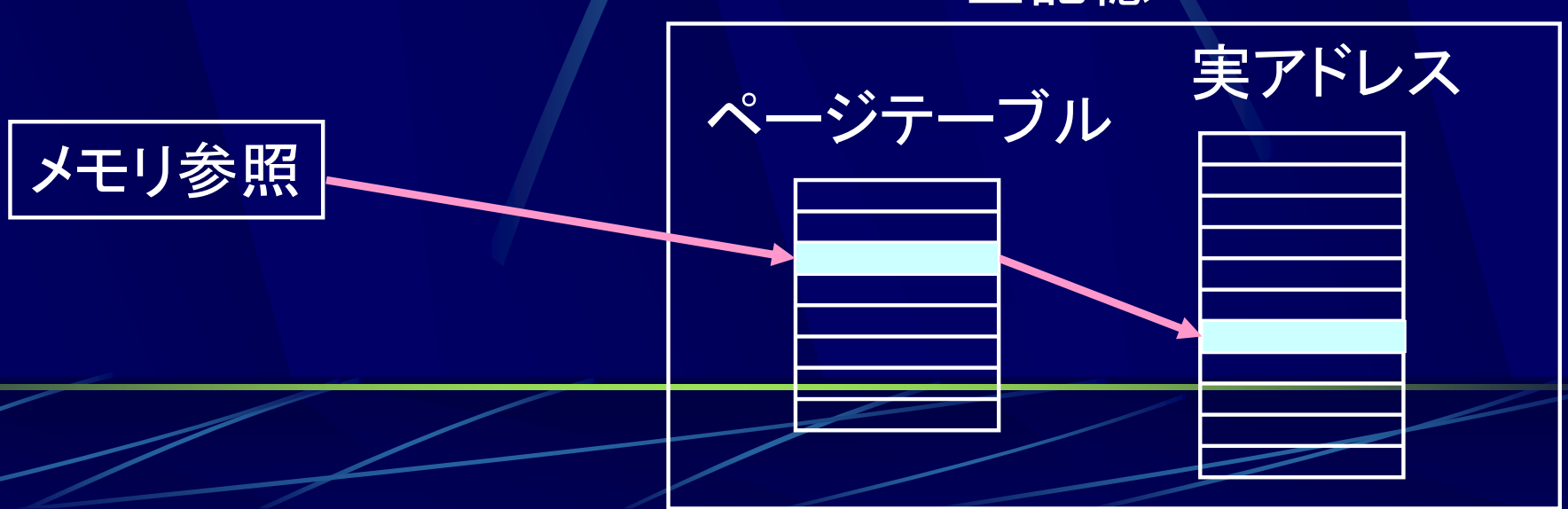
  - ページテーブルはプロセスごとに独立

  例: 100個のプロセスを実行

  ページエントリ数 約5000万
  1エントリ10Bなら 約500MB

ページングの問題点

- メモリアクセスの増大
  - ページテーブルは主記憶内に存在
  - 2回の主記憶アクセスが必要
    1. ページテーブルへのアクセス
    2. 実アドレスへのアクセス



ページングの問題点

- ページテーブルが巨大

  ハッシュ(hash)関数によるページテーブル

- メモリアクセスの増大

  連想レジスタ方式

アクセス時間

- 主記憶へのアクセス
  - 10^{-7} 秒程度
- 2次記憶へのアクセス
  - 10^{-3} 秒程度

アクセス時間に約10000倍の差

アクセス時間

主記憶へのアクセス： 10^{-7}秒
2次記憶へのアクセス：10^{-3}秒

- 主記憶上にページがある場合
  1. ページテーブルへのアクセス ：10^{-7}秒
  2. 主記憶へのアクセス ：10^{-7}秒
- 主記憶上にページが無い場合
  1. ページテーブルへのアクセス ：10^{-7}秒
  2. ページアウト ：10^{-3}秒
  3. ページイン ：10^{-3}秒
  4. 主記憶へのアクセス ：10^{-7}秒

10000倍の時間差

アクセス時間

- 主記憶上にページが無い場合
  - ページアウト, ページインに 10^{-3} 秒

↓

ページテーブルに 10^{-7} 秒でアクセスする意味が無い

↓

主記憶無いページへのエントリを
主記憶上に置く意味が無い

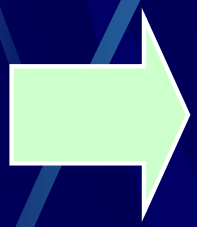
2次記憶上に置いても問題無し

ページテーブルの縮小

ページテーブル

ページ	ページ枠	フラグ		
		V	P	C
00		0	100	0
01	0	1	110	0
02		0	110	0
03	2	1	110	1
04		0	110	0
05	1	1	110	0
06	3	1	100	1
07		0	111	0

主記憶上にあるエントリのみ
テーブルに登録する



ページテーブル

ページ枠	ページ	フラグ		
		V	P	C
0	01	1	110	0
1	05	1	110	0
2	03	1	110	1
3	06	1	100	1

エントリ数 =
仮想記憶のページ数

エントリ数 =
主記憶のページ枠数

ハッシュ(hash)関数

- ハッシュ(hash)関数
  - データを一定長のデータに要約
  - 一種の圧縮
  - データの損失が起こる(不可逆)

例：数値を1桁に要約するハッシュ関数

$$h(x) = x \bmod 10$$

x	$h(x)$
2194	4
639	9
3842	2
17	7
332	2
331	1
2835	5

重複

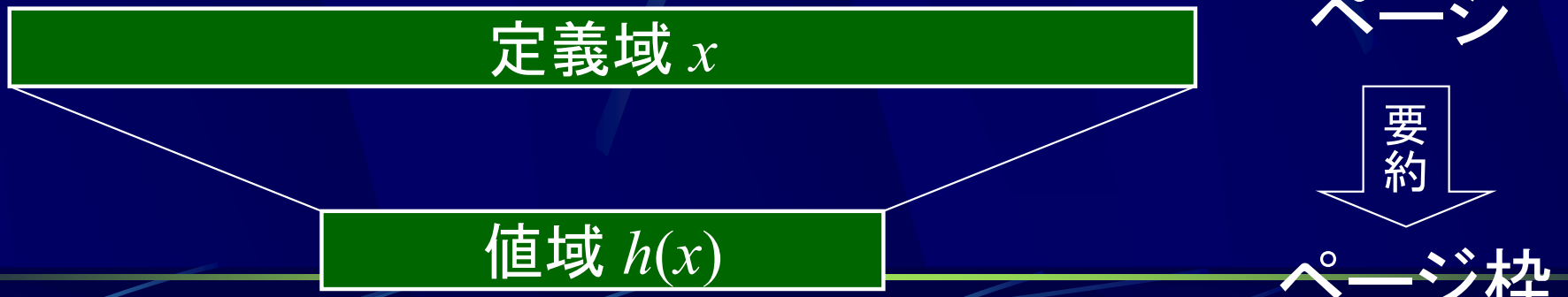
ハッシュ(hash)関数

- ハッシュ(hash)関数

ある範囲(定義域)のデータ

要約

ある範囲(値域)のデータ



ページングの動作

ハッシュ関数

$h(x) = x \bmod 4$

仮想アドレス

05	246

05 mod 4 = 1

実アドレス

1	246

主記憶

ページ枠	ページ
0	
1	05
2	
3	

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
07	0F

基本的には
ハッシュ関数の
値が実アドレス

しかし値域の重複がある
(01, 05, 09, 0D が同じ値域)

ページングの動作

主記憶

ページ枠	ページ
0	
1	05
2	
3	

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
07	0F

インデックスに
ハッシュ関数を使用

ページテーブル

$h(x)$	ページ	ページ枠	ポインタ
0			
1	05	1	
2			
3			

主記憶と
同じサイズ

ページングの動作

主記憶上にある場合

仮想アドレス

05	333

実アドレス

1	333

05 mod 4

$h(x)$	ページ	ページ枠	ポインタ
0			
1	05	1	
2			
3			

主記憶

ページ枠	ページ
0	
1	05
2	
3	

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
07	0F

ページが一致
＝主記憶上に有り

ページングの動作

主記憶上に無い場合

仮想アドレス

01	555

01 mod 4

$h(x)$	ページ	ページ枠	ポインタ
0			
1	05	1	
2			
3			

主記憶

ページ枠	ページ
0	
1	05
2	
3	

ページ枠2に
ページインしたい

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
07	0F

ページが不一致
＝主記憶上に無し

ページングの動作

主記憶上に無い場合

仮想アドレス

01	555

実アドレス

2	555

01 mod 4

$h(x)$	ページ	ページ枠	ポインタ
0			
1	05	1	2
2	01	2	
3			

主記憶

ページ枠	ページ
0	
1	05
2	01
3	

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
	0F

エントリへのポインタを記録しておく

ページングの動作

主記憶上に無い場合

仮想アドレス

09	777

09 mod 4

$h(x)$	ページ	ページ枠	ポインタ
0			
1	05	1	2
2	01	2	
3			

ページが不一致

主記憶

ページ枠	ページ
0	
1	05
2	01
3	

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
07	0F

ページングの動作

主記憶上に無い場合

仮想アドレス

09	777

実アドレス

3	777

09 mod 4

$h(x)$	ページ	ページ枠	ポインタ
0			
1	05	1	2
2	01	2	3
3	09	3	

主記憶

ページ枠	ページ
0	
1	05
2	01
3	09

2次記憶

ページ	
00	08
01	09
02	0A
03	0B
04	0C
05	0D
06	0E
	0F

エントリが見つかるまで
ポインタを辿っていく

ハッシュ関数の条件

- 値域への均等分布
  - ハッシュ関数による変換結果に偏りが無い
    - 偏りがあると表の1箇所にエントリが集中

    ⇒ 検索時にポインタを辿る確率が高くなる
- 高速に変換可能
  - ハッシュ関数はアクセスのたびに計算

  ⇒ 変換に時間が掛かると意味が無い

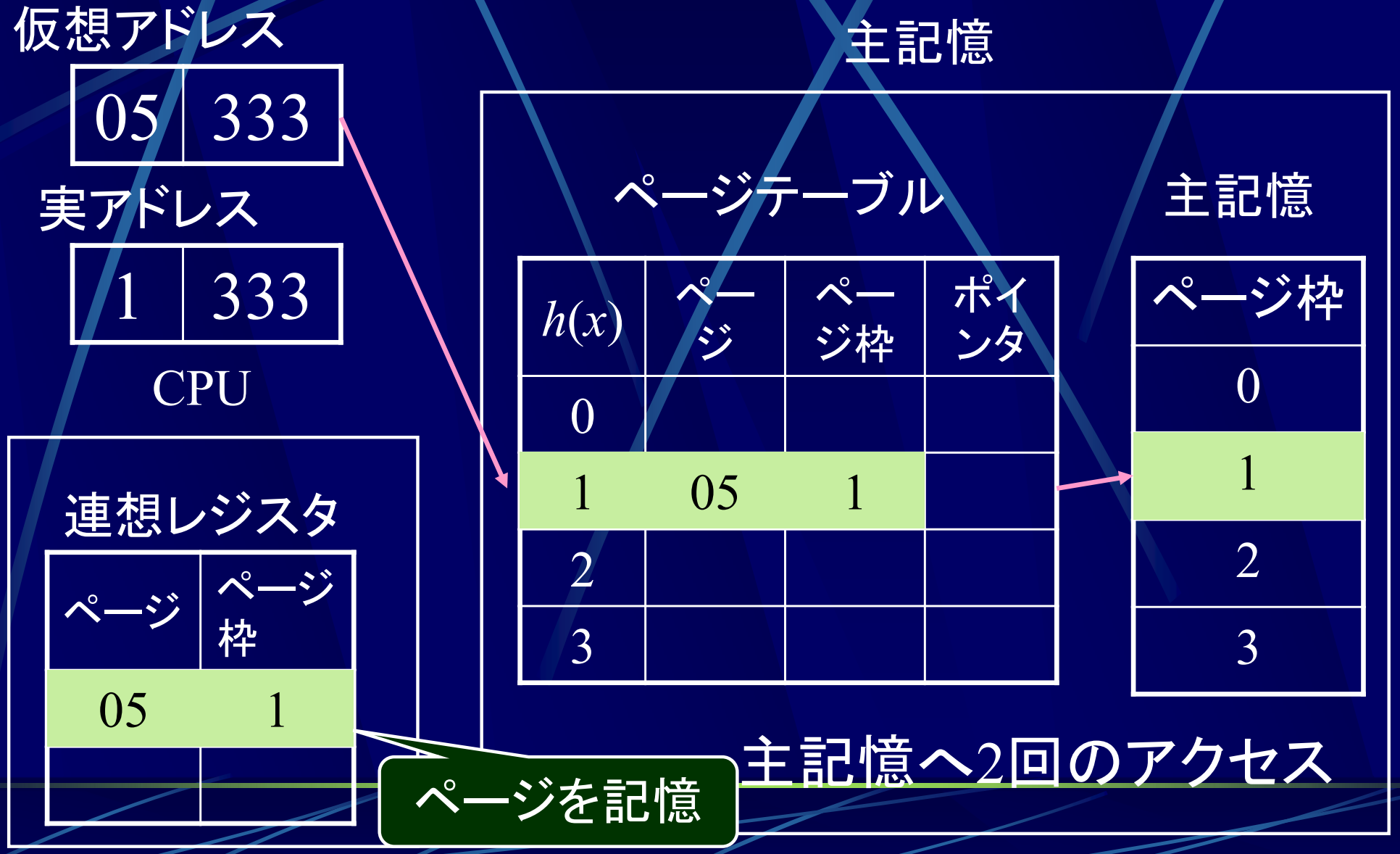
連想レジスタ

一般的にプログラムは
一度アクセスしたアドレスを近いうちに
再アクセスすることが多い

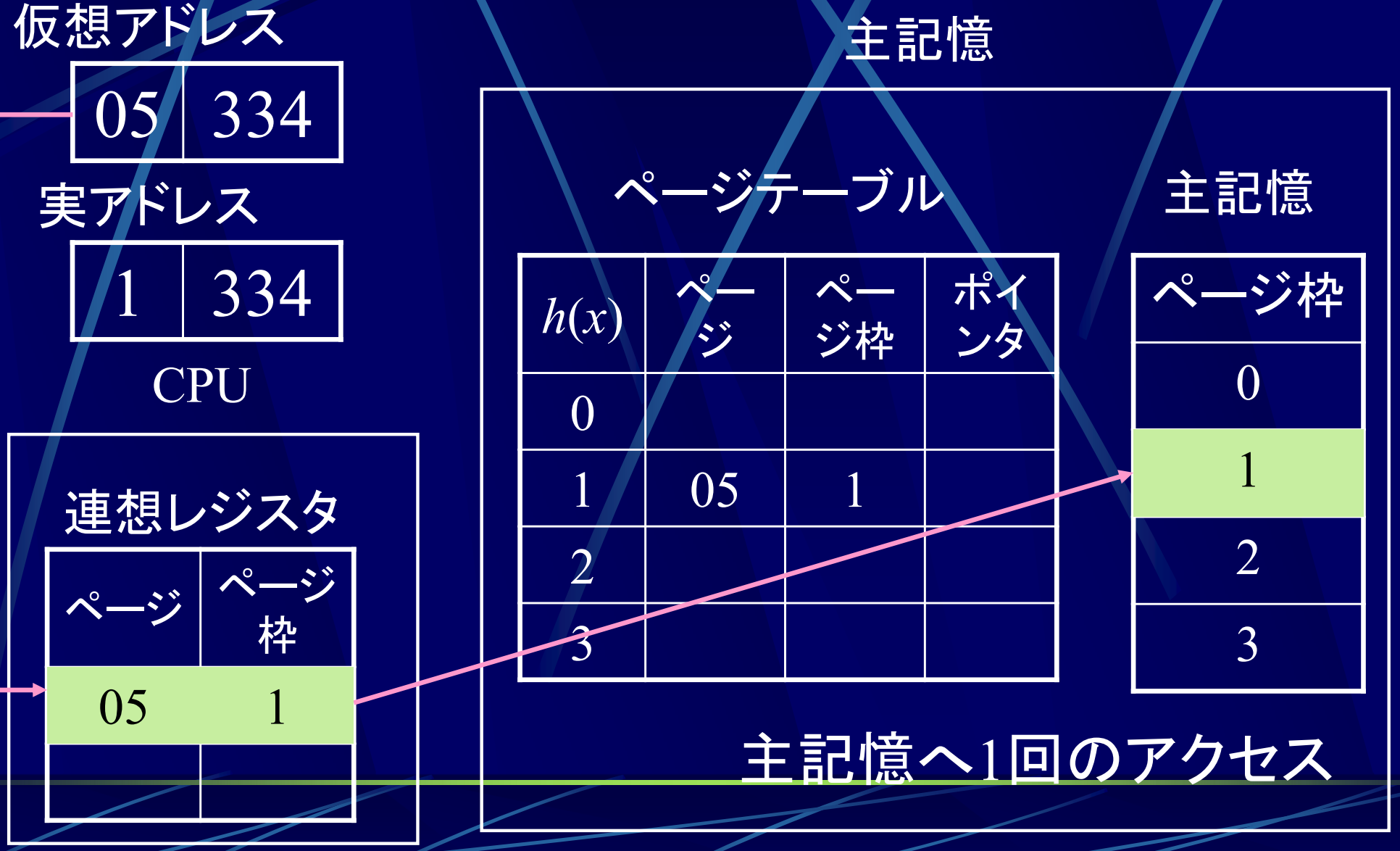
最近参照されたページテーブルをCPU内で記憶

連想レジスタ
- 小容量
- 高速

連想レジスタ


仮想アドレス
05
333
実アドレス
1
333
CPU
連想レジスタ
ページ
ページ枠
05
1
主記憶
ページテーブル
h(x)
ページ
ページ枠
ポインタ
0
1
05
1
2
3
主記憶
ページ枠
0
1
2
3
ページを記憶
主記憶へ2回のアクセス

連想レジスタ


仮想アドレス
05
334
実アドレス
1
334
CPU
連想レジスタ
ページ
ページ枠
05
1
主記憶
ページテーブル
h(x)
ページ
ページ枠
ポインタ
0
1
05
1
2
3
主記憶
ページ枠
0
1
2
3
主記憶へ1回のアクセス

連想レジスタ

- 主記憶へのアクセス回数
  - 初めてアクセスするページ　：2回
  - 最近アクセスしたベージ　：1回

統計的にはレジスタが8～16 個で90%のヒット率

まとめ

- 仮想記憶
  - 2次記憶を用いて主記憶よりも大きな記憶領域を確保
  - 主記憶上に無いデータはスワップイン
- ページング
  - 主記憶の再配置システム(非連続割り付け)
  - 仮想アドレスの上位(ページ番号)を実アドレスの上位(ページ枠番号)に変換

まとめ:ページングの長所と短所

- 長所
  - 外部断片化が起きない
  - 内部断片化で無駄になる領域は少ない
- 短所
  - ページテーブルが巨大
  - メモリアクセスの増大

まとめ:ページングの問題点の解法

- ページテーブルが巨大
  - ハッシュ(hash)関数によるページテーブル

テーブルサイズ
ページ数(仮想記憶) ⇒ ページ枠数(実記憶)

- メモリアクセスの増大
  - 連想レジスタ方式

主記憶へのアクセス回数
2回 ⇒ 最近使用したページは1回

統計的に90%ヒット