オペレーティングシステム

第8回

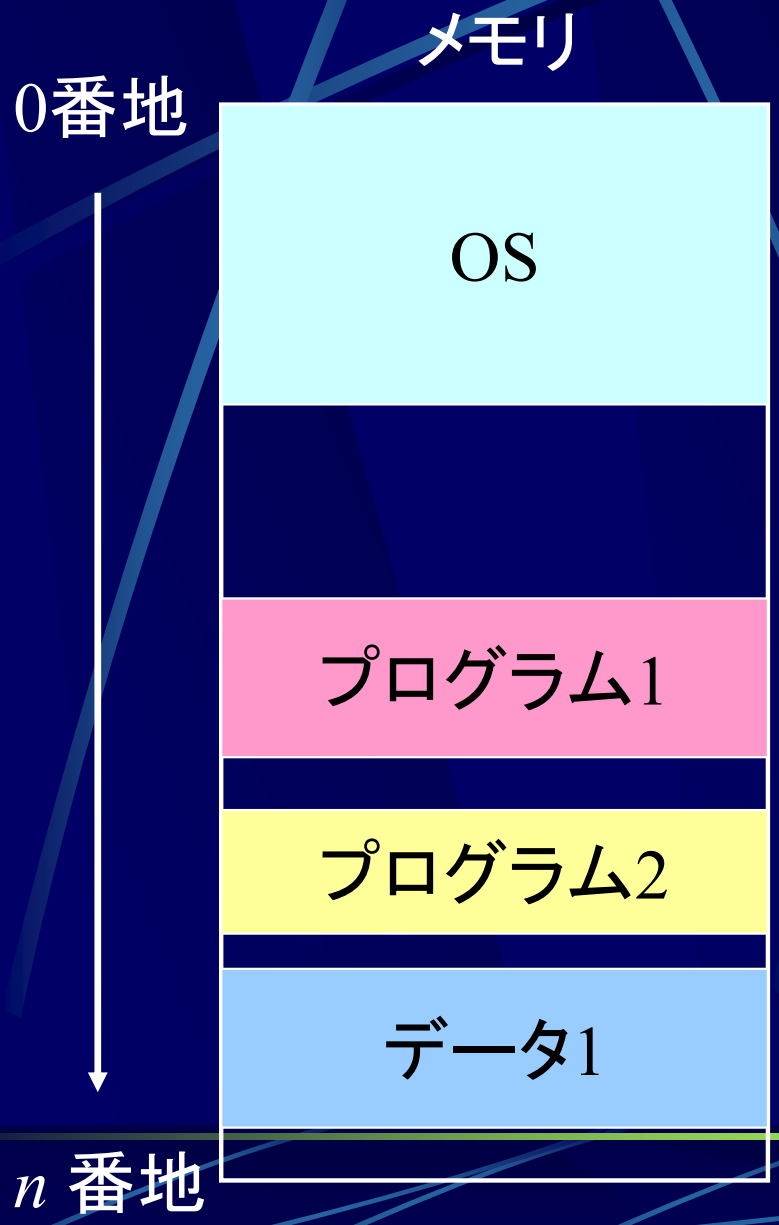
実記憶管理

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

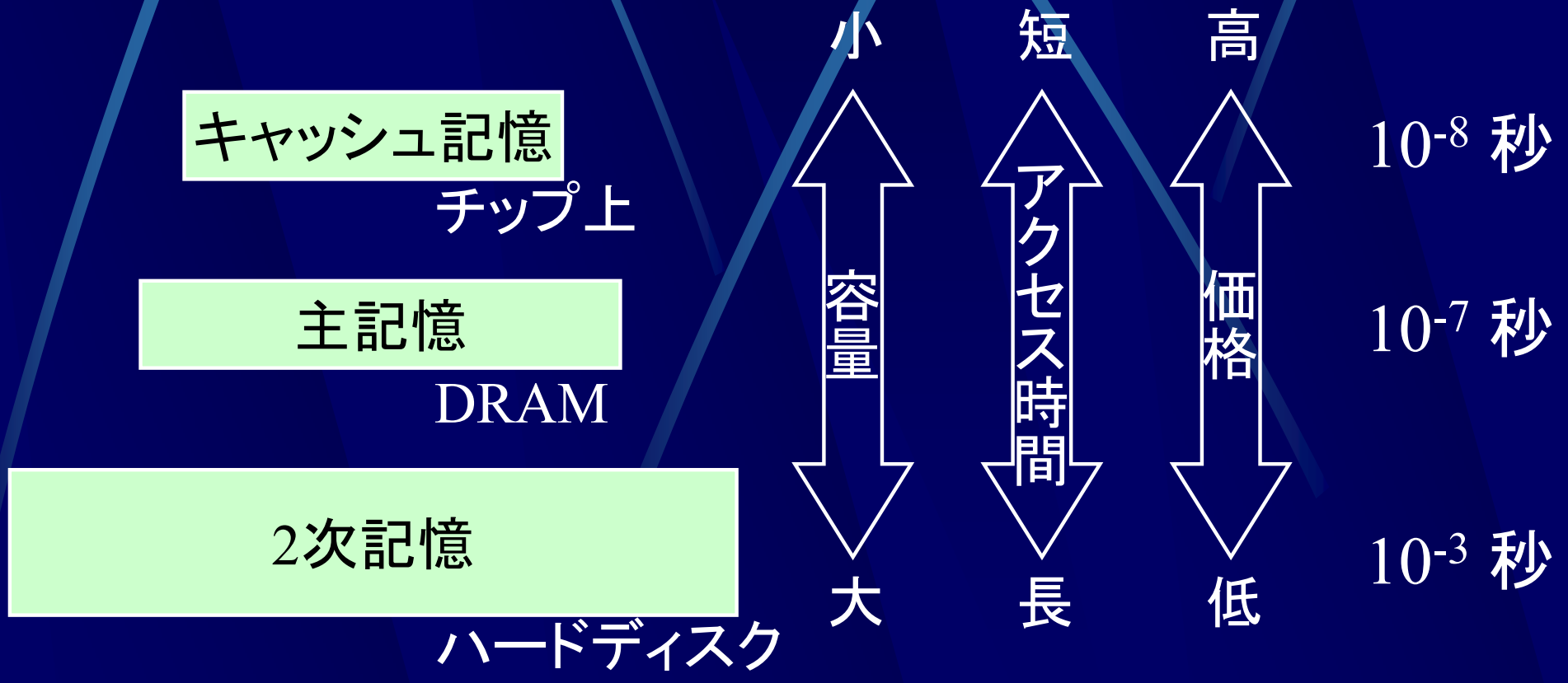


OS, ユーザプログラム, データ
メモリ上に置かれる

メモリ上の位置は
1 次元のアドレスで管理

メモリ

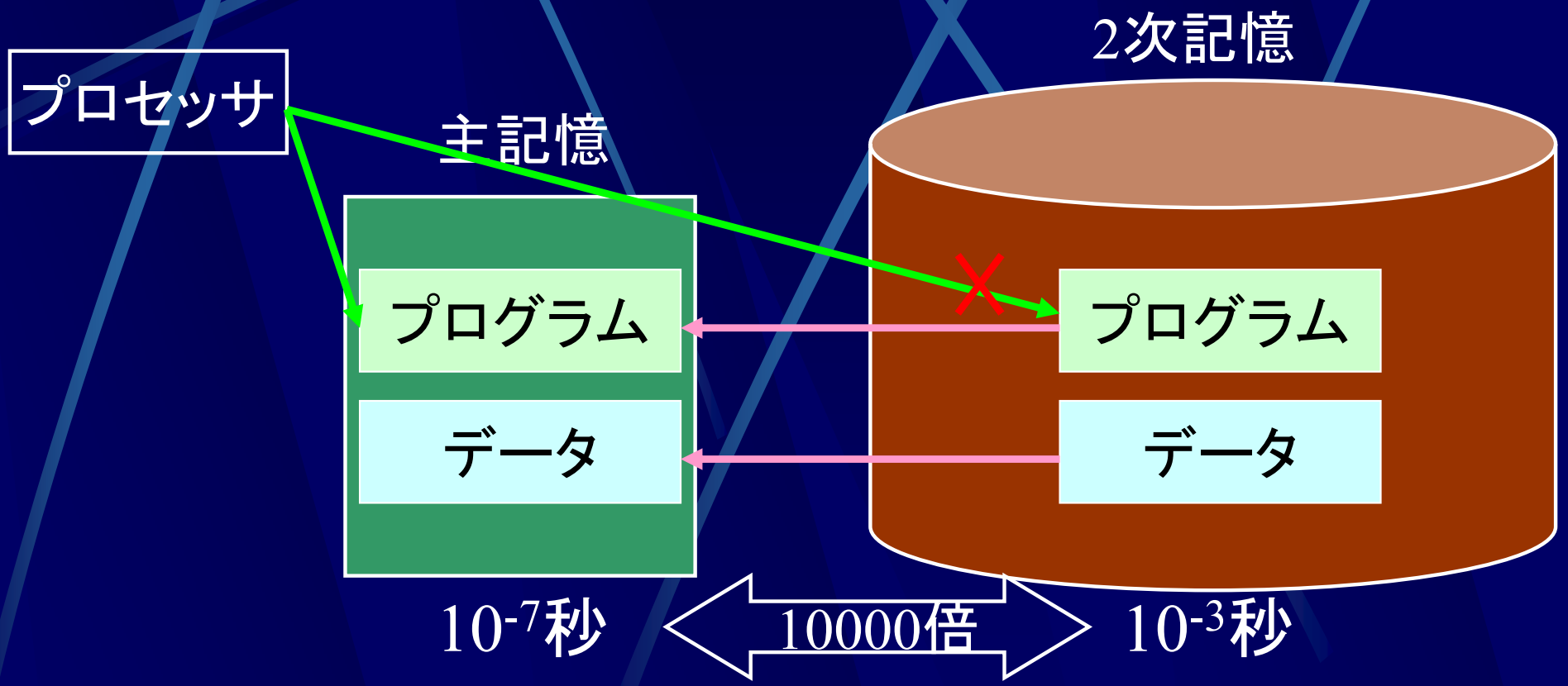
- メモリの記憶階層



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

メモリ管理技法

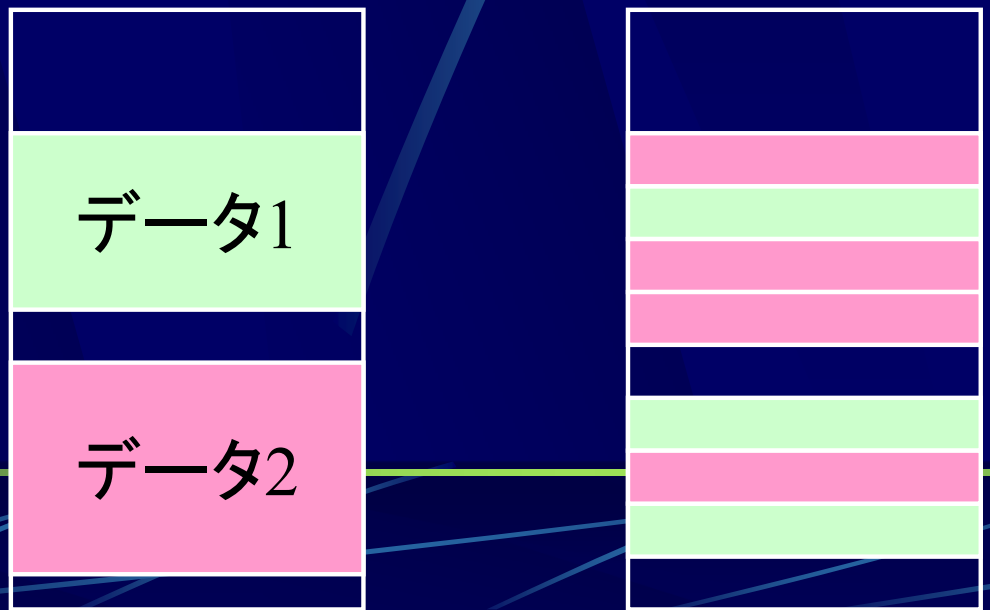
- メモリ管理技法
  - 割り付け技法(placement)
    - プログラム, データのメモリ上への割り付け位置を決定
  - フェッチ技法(fetch)
    - プログラム, データを2次記憶から主記憶への読み込み時期を決定
  - 置き換え技法(replacement)
    - 空き領域作成のために2次記憶に追い出すデータの決定

割り付け技法(placement)

メモリのどこに割り当てる？

割り付け技法(placement)

- 割り付け技法
  - 連続割り付け(contiguous allocation)
    - プログラム, データをメモリ上の連続した領域に置く
  - 非連続割り付け(noncontiguous allocation)
    - プログラム, データをメモリ上に分割して置く

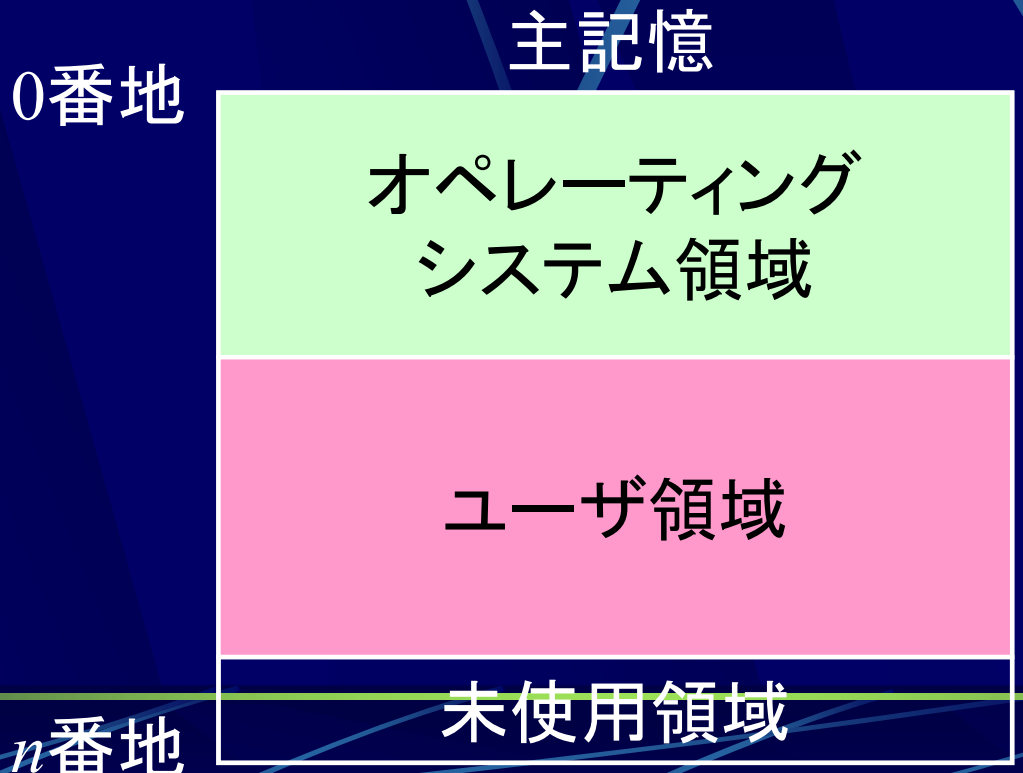


割り付け技法

<table>
<tr><td rowspan="3">連続割り付け
(contiguous allocation)</td><td>単一連続割り付け
(single partition allocation)</td><td>単一ユーザ</td></tr>
<tr><td>固定区画割り付け
(static partition allocation)</td><td rowspan="4">複数ユーザ</td></tr>
<tr><td>可変区画割り付け
(dynamic partition allocation)</td></tr>
<tr><td rowspan="2">非連続割り付け
(noncontiguous allocation)</td><td>ページング
(paging)</td></tr>
<tr><td>セグメンテーション
(segmentation)</td></tr>
</table>

単一連続割り付け

- OS領域とユーザ領域の2つに分割
- 単一のユーザのみにメモリを割り付ける



単一連続割り付け管理技法

- 単一連続割り付け管理技法
  - 再配置(relocation)
    - 相対番地で記述されたプログラムを主記憶の任意に位置に配置する
  - スワッピング(swapping)
    - 待ち状態のプログラムを2次記憶に退避させる
  - オーバレイ(overlay)
    - プログラムの必要な部分のみを主記憶上に読み込む

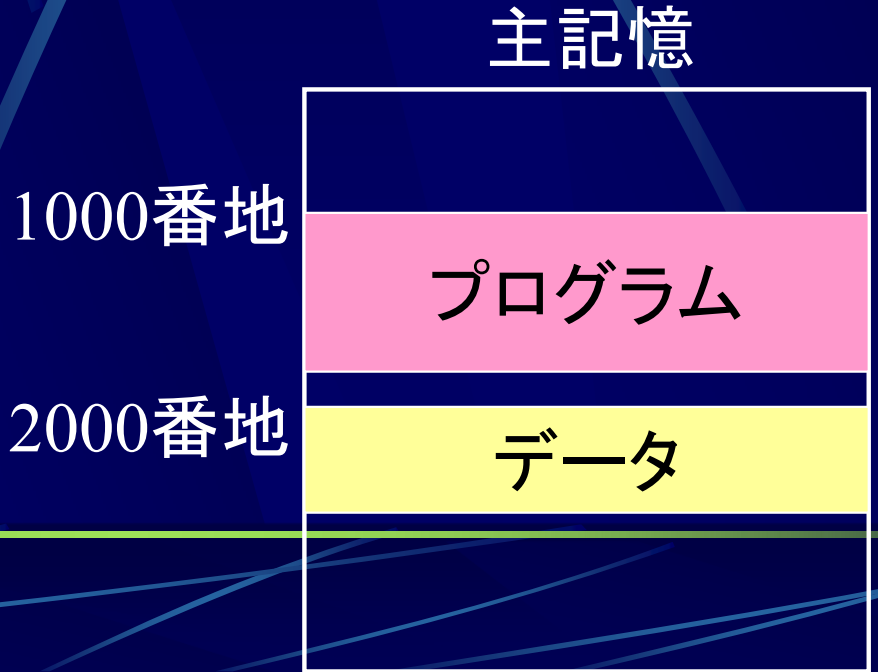
絶対番地, 相対番地
(absolute address, relative address)

- 絶対番地式(absolute address)プログラム
  - アドレスが固定されたプログラム
- 相対番地式(relative address)プログラム
  - 自身の先頭番地を0としてそこからの相対番地で記述されたプログラム

絶対番地式プログラム

- 絶対番地式プログラム
  - メモリの実アドレスでプログラムを記述
  - 必ずメモリのその位置に読み込む必要あり

```
1000 LOAD  2000
1001 LOAD  2001
1002 ADD
1003 BEQ     1010
1004 INPUT
1005 STORE 2002
        :
```

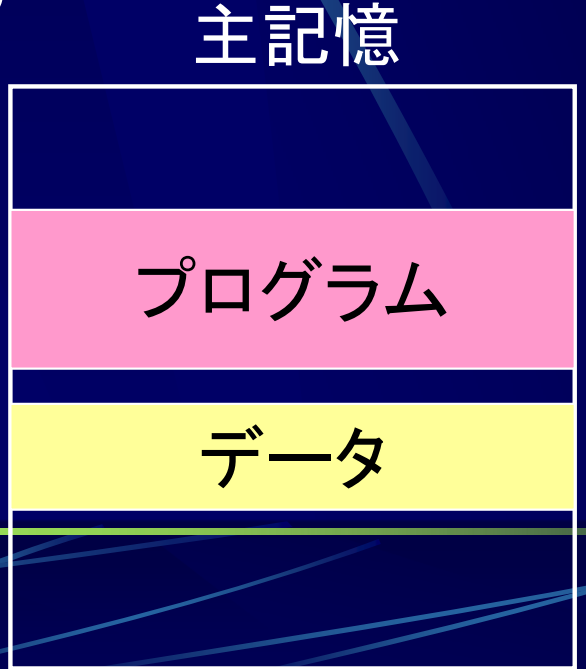


相対番地式プログラム

- 相対番地式プログラム
  - プログラムの先頭を0番地として相対的に記述
  - メモリへの読み込み時にアドレスを再計算

再配置(relocation)

```
0 LOAD  1000    1300
1 LOAD  1001    1301
2 ADD
3 BEQ       10  310
4 INPUT
5 STORE 1002    1302
      :
```



再配置(relocation)

- 再配置(relocation)
  - 静的再配置(static relocation)
    - メモリに読み込む際に結合(binding)
    - OS領域とユーザ領域の境界アドレスが変更されると再読み込みが必要
  - 動的再配置(dynamic relocation)
    - 実行時に再配置
    - 1度読み込めば再読み込みの必要は無い
    - 再配置レジスタ(relocation register)が必要

静的再配置

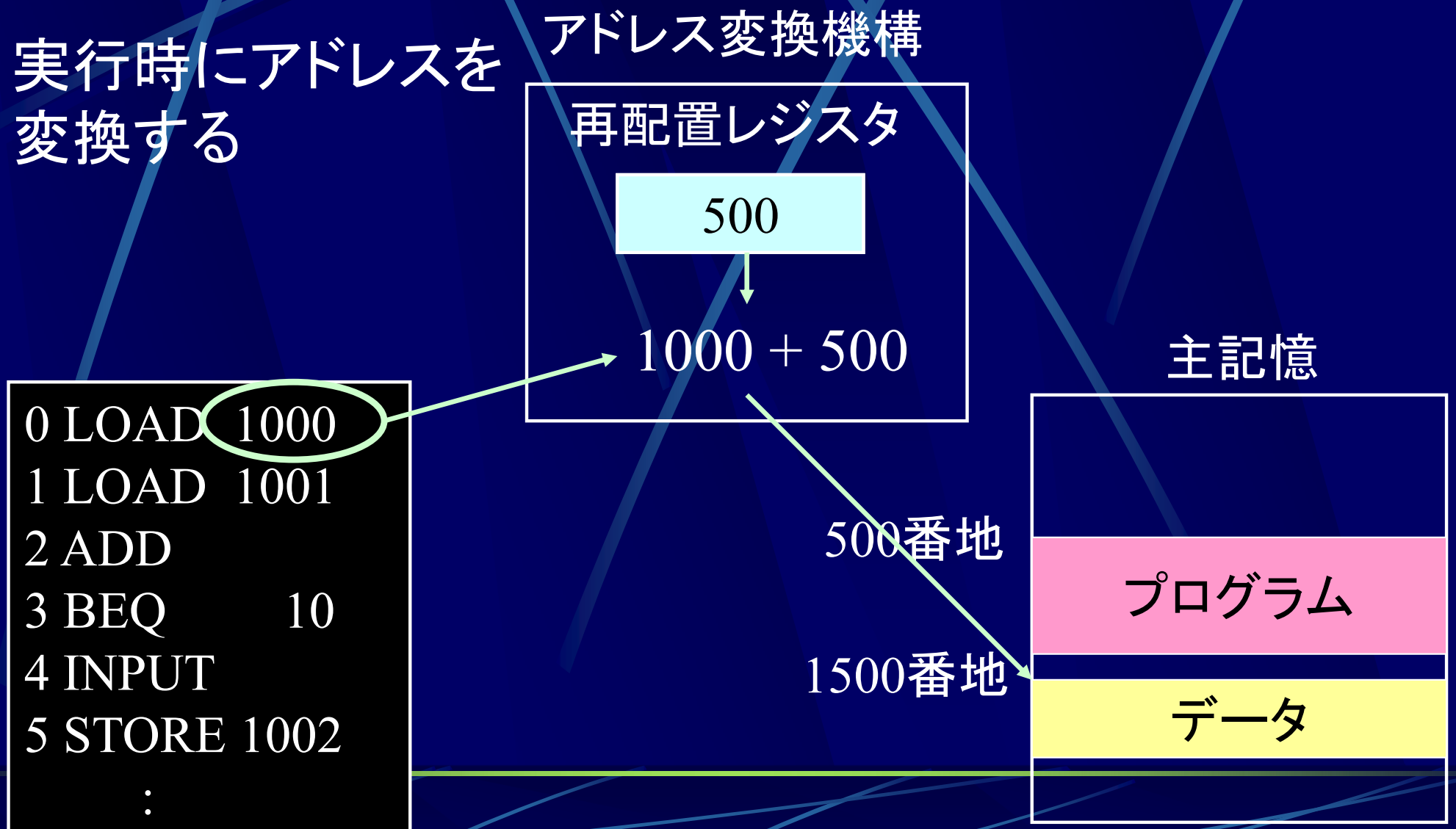
読み込み時にアドレスを変換する

動的再配置

読み込み時はアドレスはそのまま

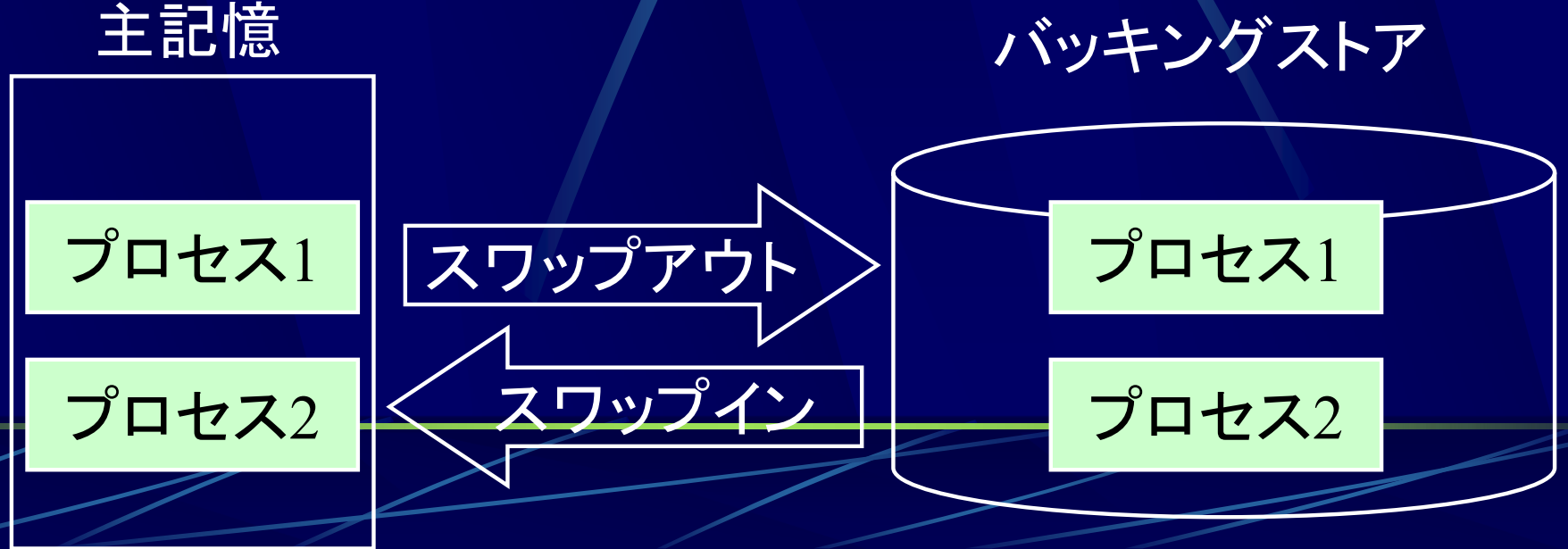
動的再配置

実行時にアドレスを
変換する



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

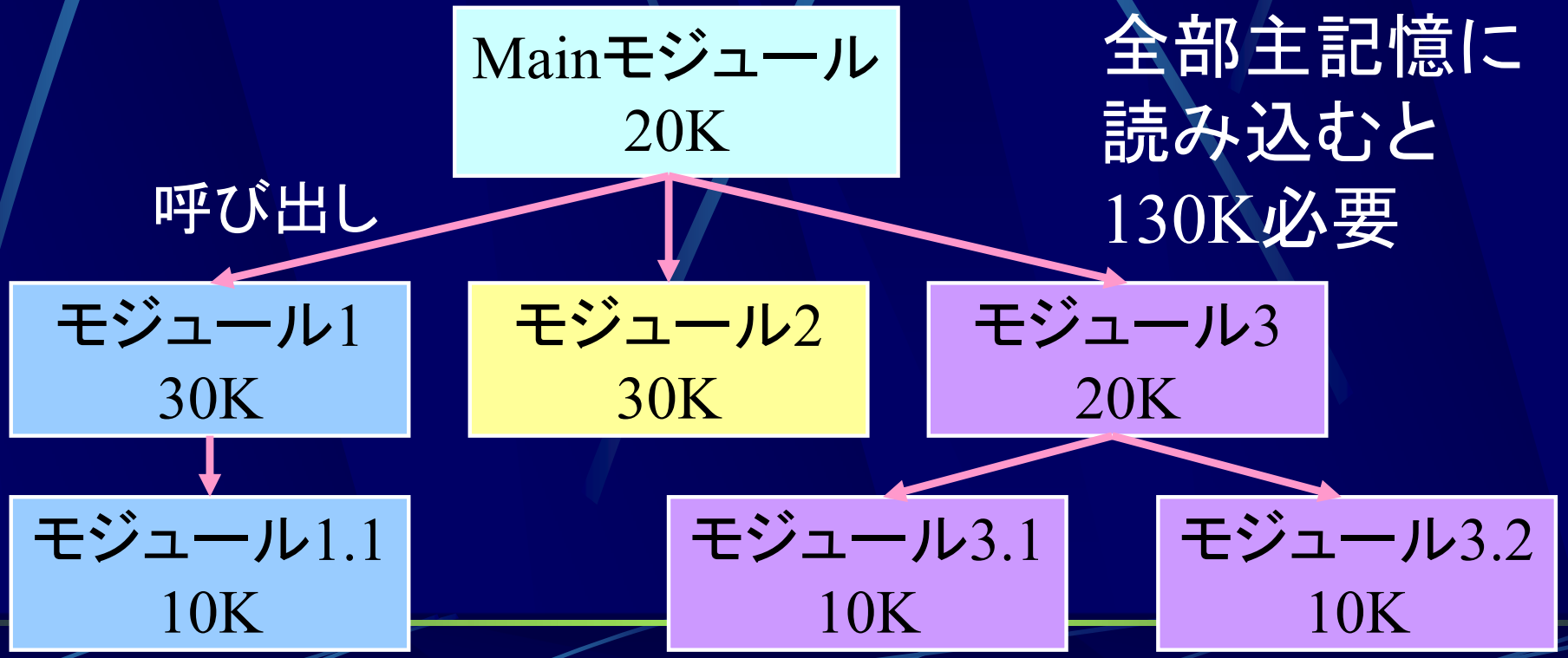


スワップイン, スワップアウト
(swap-in, swap-out)

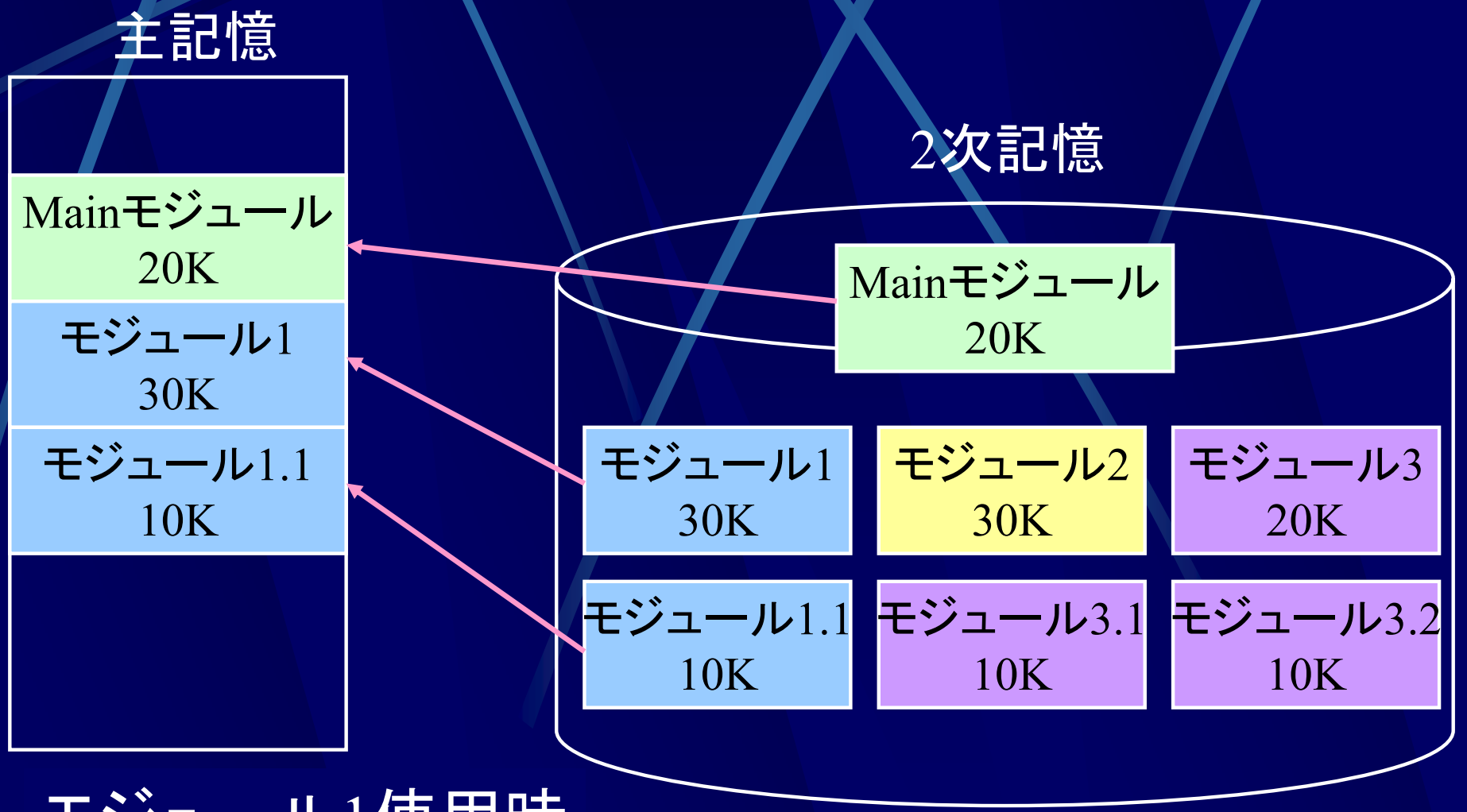
- スワップイン(swap-in)
  - プログラム, データを2次記憶から主記憶に
  - 実行に必要なものを読み込む
- スワップアウト(swap-out)
  - プログラム, データを主記憶から2次記憶に
  - スワップインの領域を確保するために当面必要の無いものを退避させる

オーバレイ(overlay)

- オーバレイ(overlay)
  - プログラムの必要部分のみを読み込む

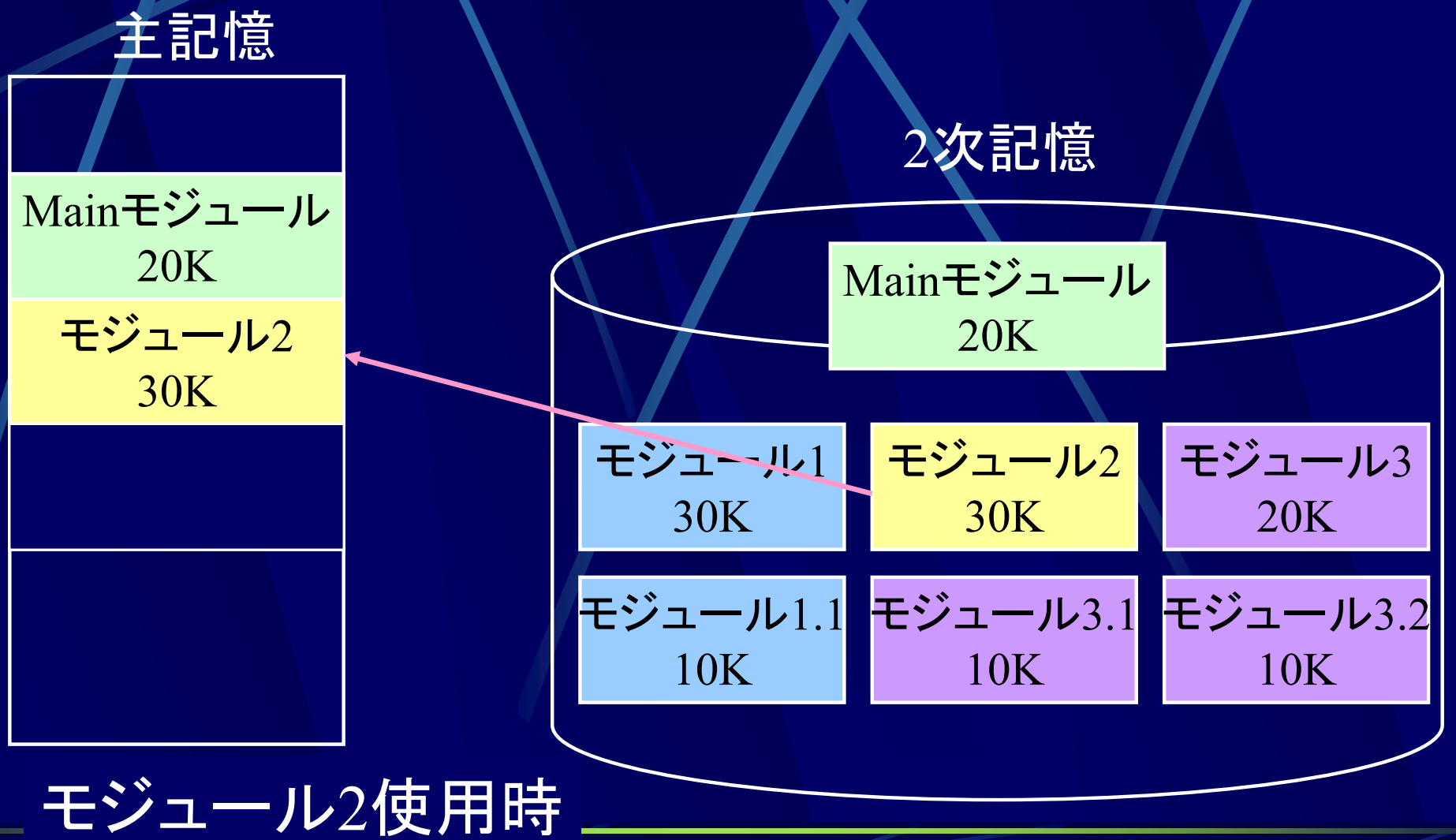


オーバレイ



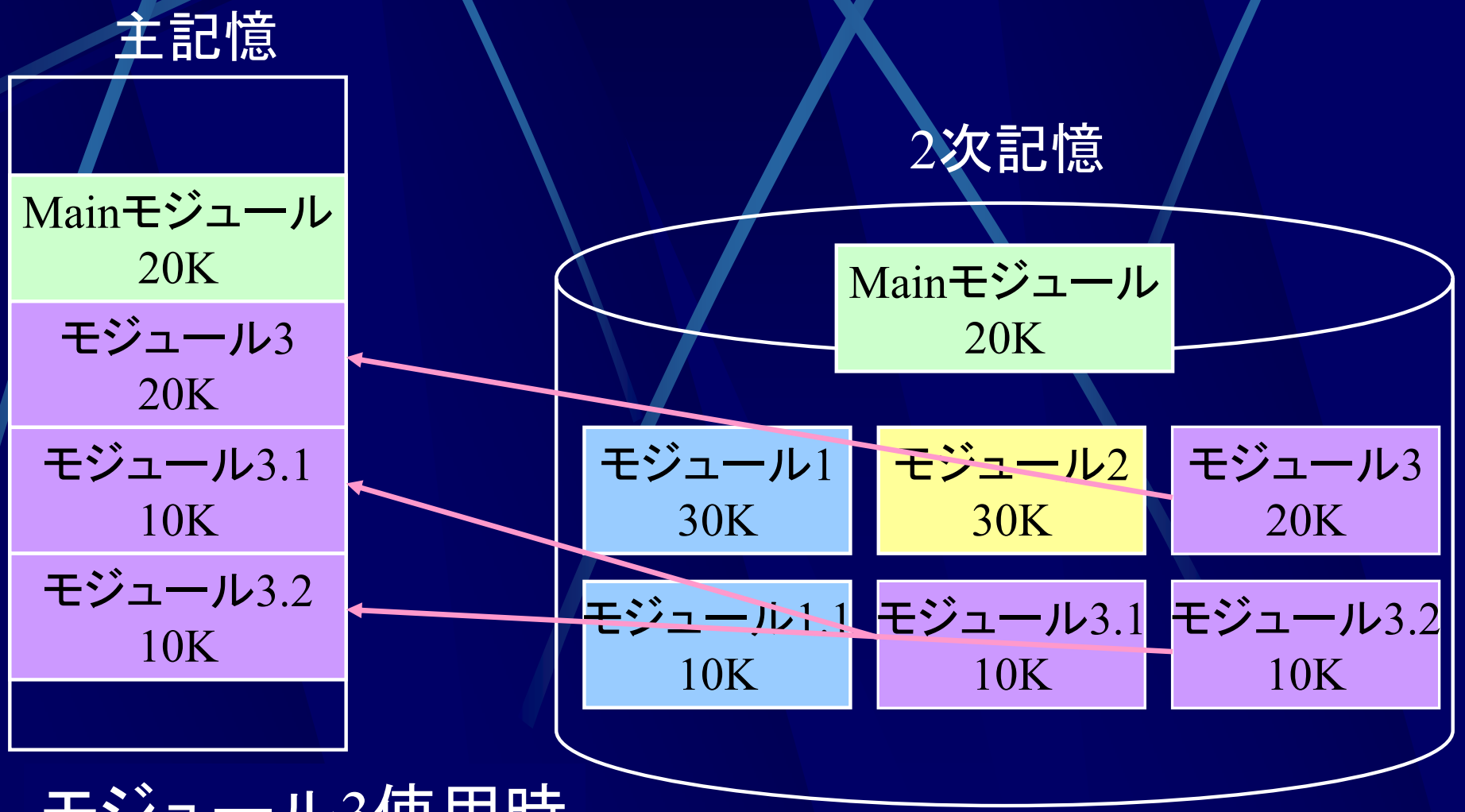
モジュール1使用時

オーバレイ



モジュール2使用時

オーバレイ



モジュール3使用時

主記憶は60Kあればいい

区画割り付け

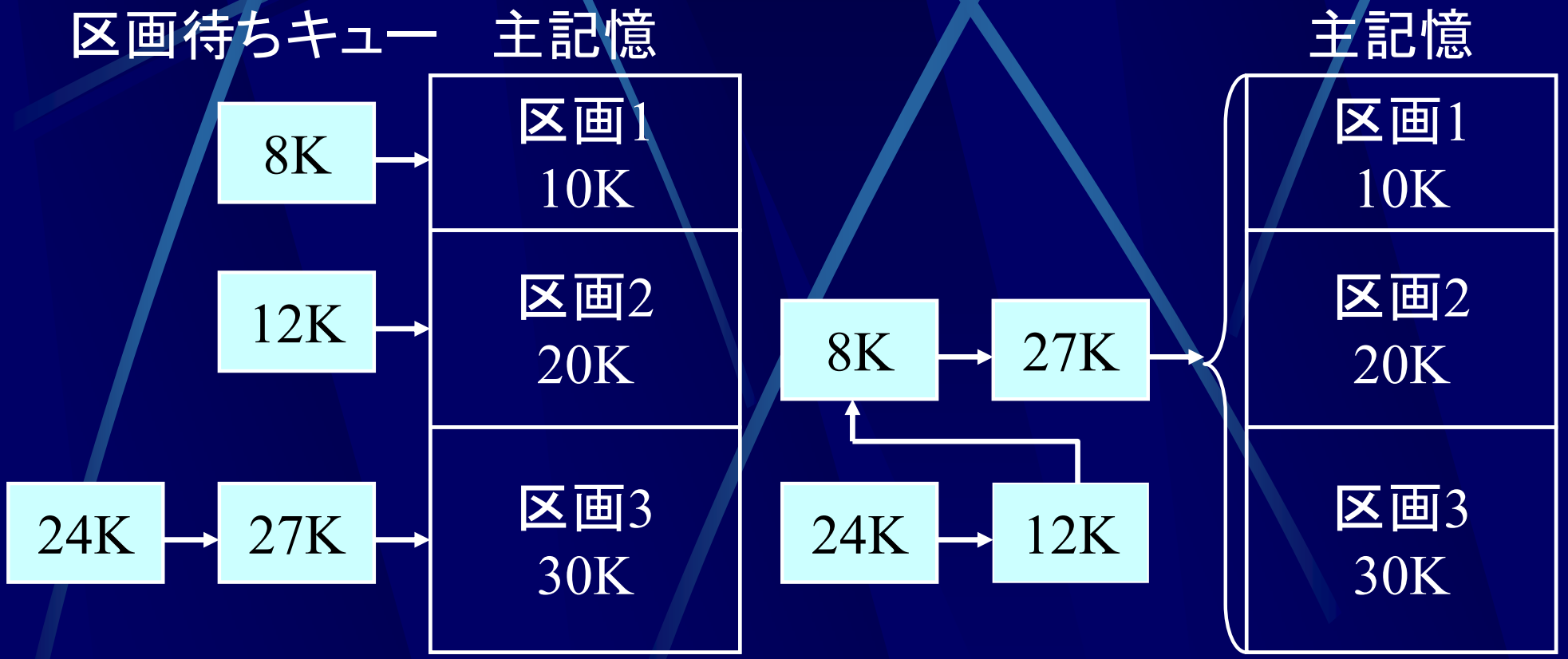
- 固定区画割り付け(static partition allocation)
  - 区画の大きさは予め決定, プロセスが必要とするサイズ以上の区画に割り付け
- 可変区画割り付け(dynamic partition allocation)
  - 区画の大きさをプロセスに応じて変更
- バディシステム(buddy system)
  - プロセスが必要とするサイズ以上のサイズ 2^k の区画を割り付け

固定区画割り付け
(static partition allocation)

- 固定区画割り付け
  - OSにより主記憶が予め決まった大きさの区画に分割

区画1には 10K 以下のプログラム, データしか読み込めない

固定区画割り付け



必要サイズ以上で空いている区画に割り当て可能

固定区画割り付け

- 絶対番地式
  - 区画ごとに待ちキューを設定
  - キュー間で競合が起きない
- 相対番地式
  - 区画全体を1つのキューで管理
  - スケジューリングアルゴリズムで割り当てる区画を決定

相対番地式のスケジューリング

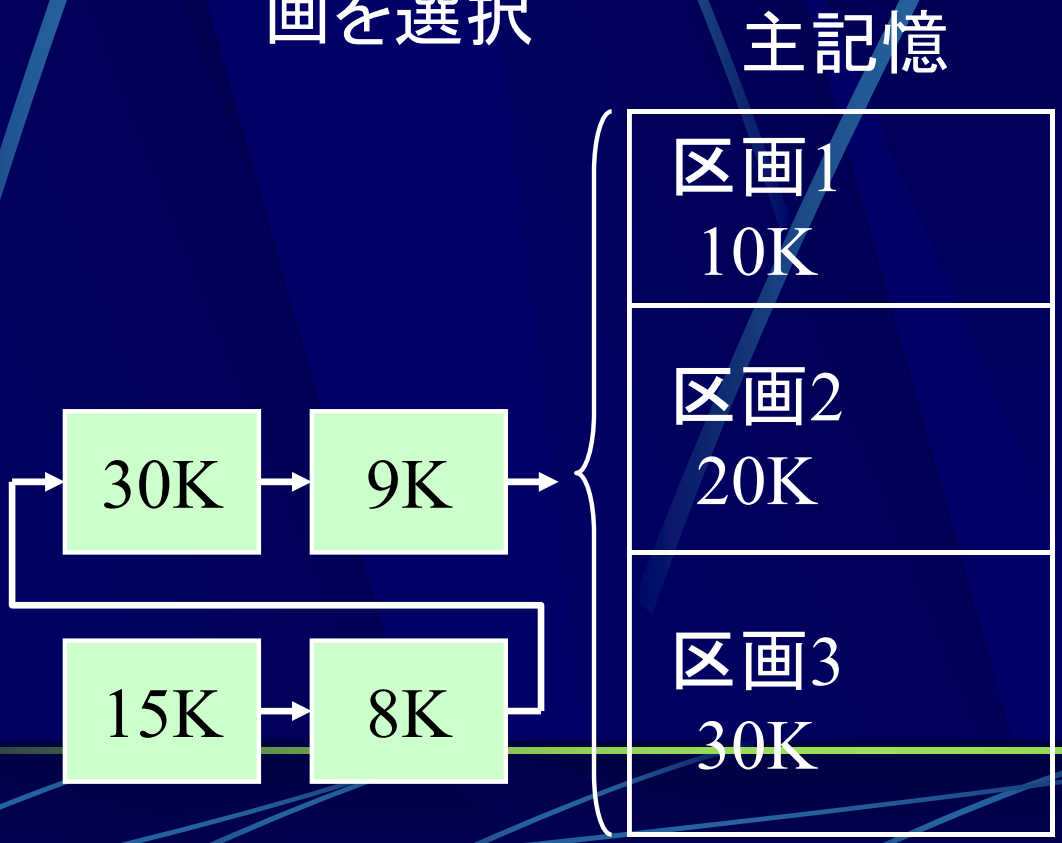
- 静的再配置とFCFSスケジューリング
  - 到着順で配置する
- 静的再配置とスワッピング
  - 優先度に基づいてスワッピング
  - スワップアウトしたプログラムは同じ区画へ
- 動的再配置とスワッピング
  - 優先度に基づいてスワッピング
  - スワップアウトしたプログラムは区画を再選択

静的再配置と FCFSスケジューリング

- 静的再配置とFCFS スケジューリング
  - 方法1 : 最小空き区画選択
    - 必要とする大きさ以上の**空き区画**の中で最小の区画を選択
  - 方法2 : 最小区画選択
    - 必要とする大きさ以上の**区画**(空き, 使用中問わず)の中で最小の区画を選択

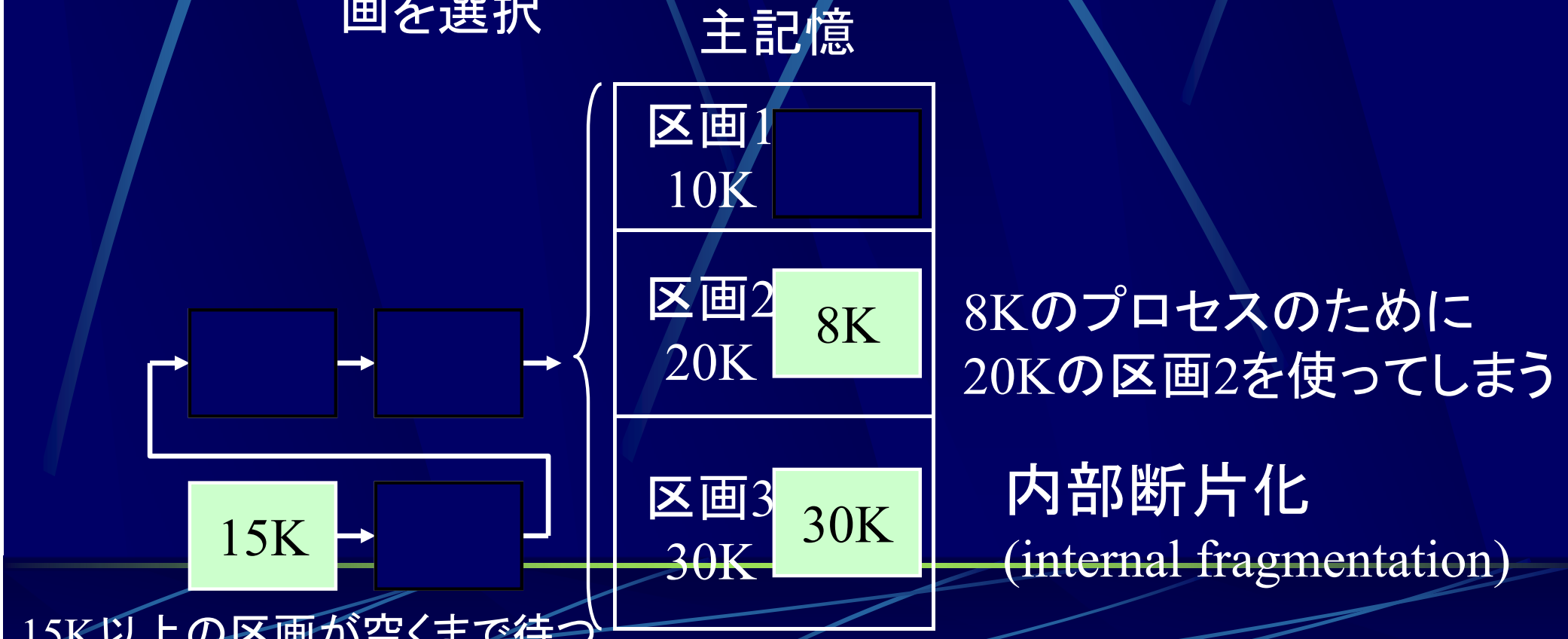
静的再配置と FCFSスケジューリング

- 最小空き区画選択
  - 必要とする大きさ以上の空き区画の中で最小の区画を選択



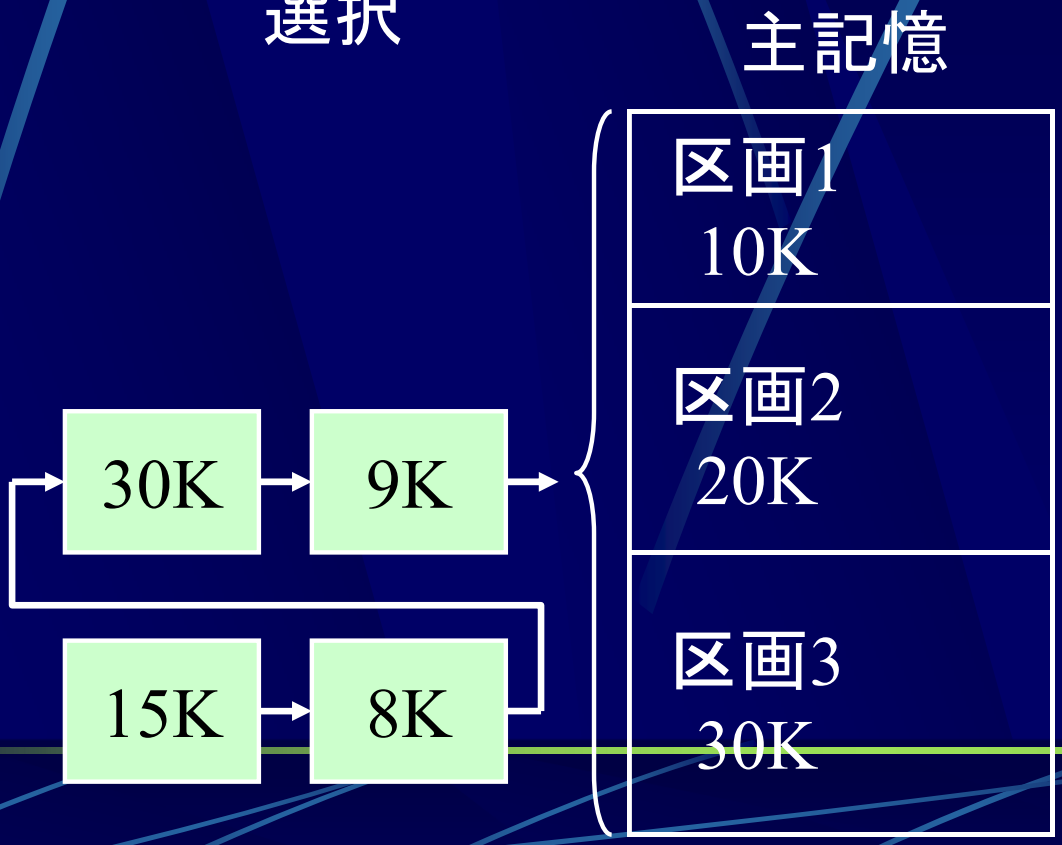
静的再配置とFCFSスケジューリング

- 最小空き区画選択
  - 必要とする大きさ以上の空き区画の中で最小の区画を選択



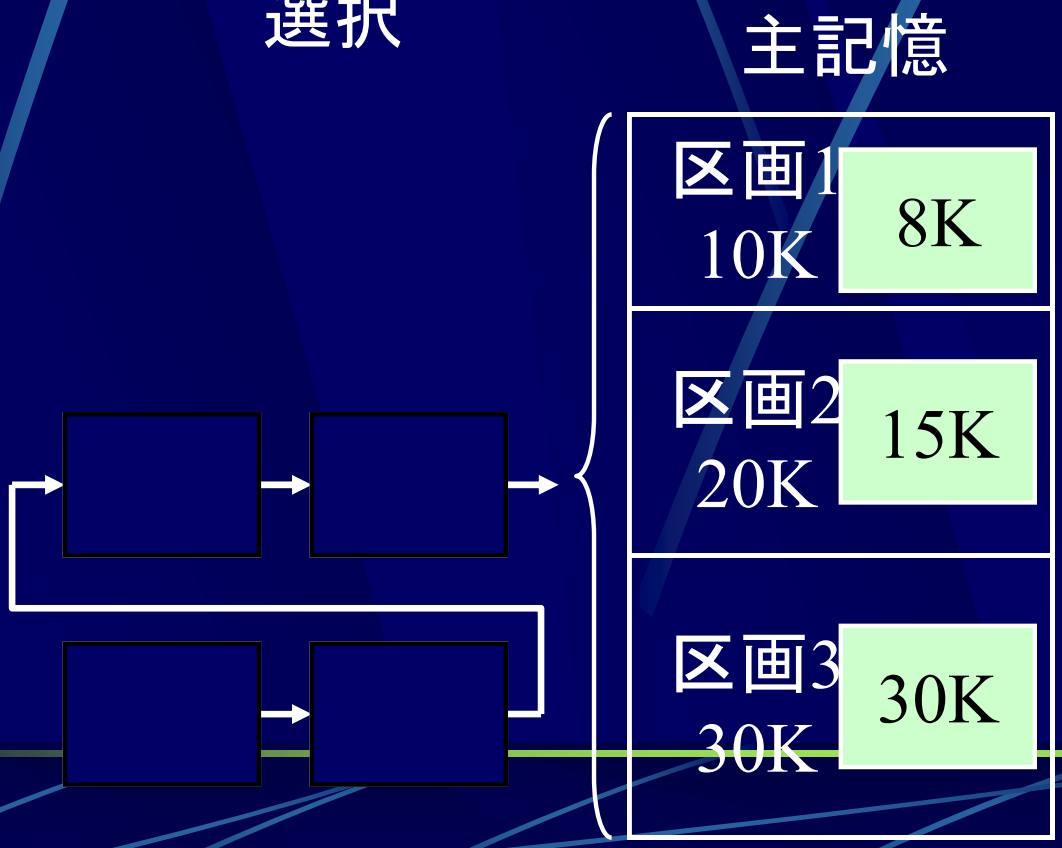
静的再配置と FCFSスケジューリング

- 最小区画選択
  - 必要とする大きさ以上の区画の中で最小の区画を選択



静的再配置と FCFSスケジューリング

- 最小区画選択
  - 必要とする大きさ以上の区画の中で最小の区画を選択



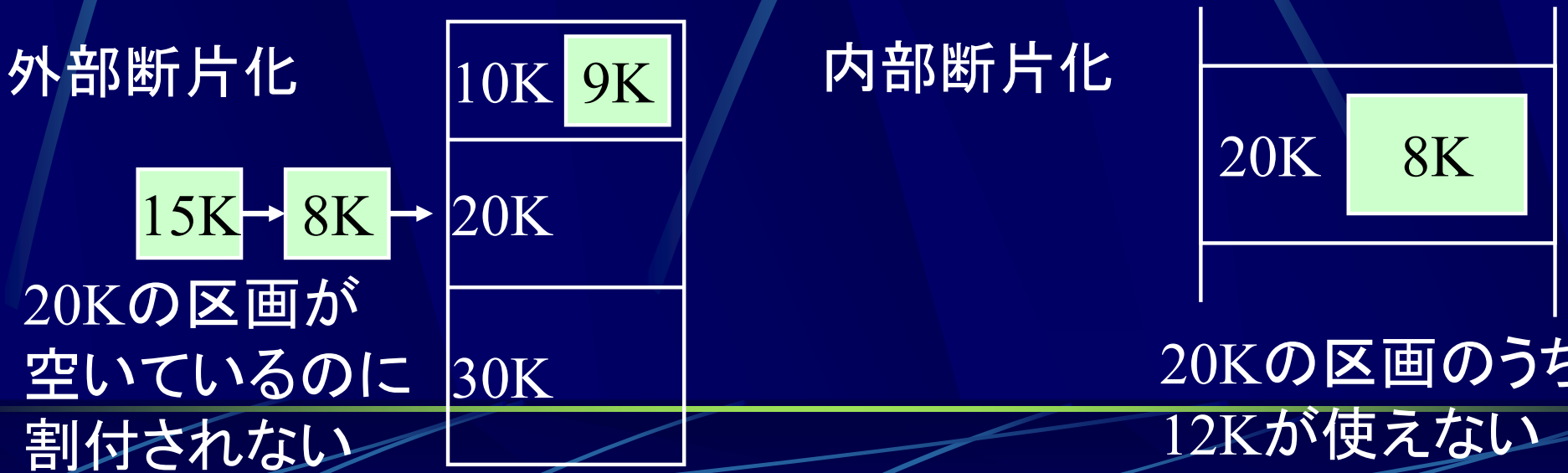
8Kのプロセスは
9Kのプロセスが
終わった後に
割り付け

区画2が空いているのに
割り付けされない

外部断片化
(external fragmentation)

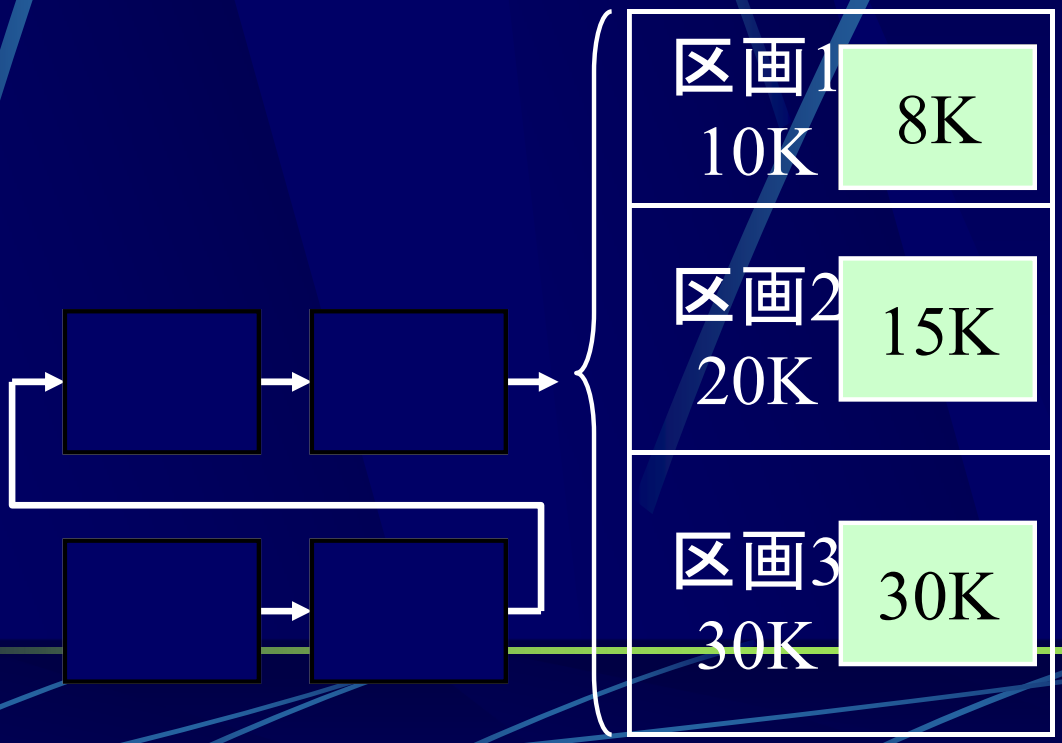
外部断片化, 内部断片化
(external fragmentation, internal fragmentation)

- 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



静的再配置と FCFSスケジューリング

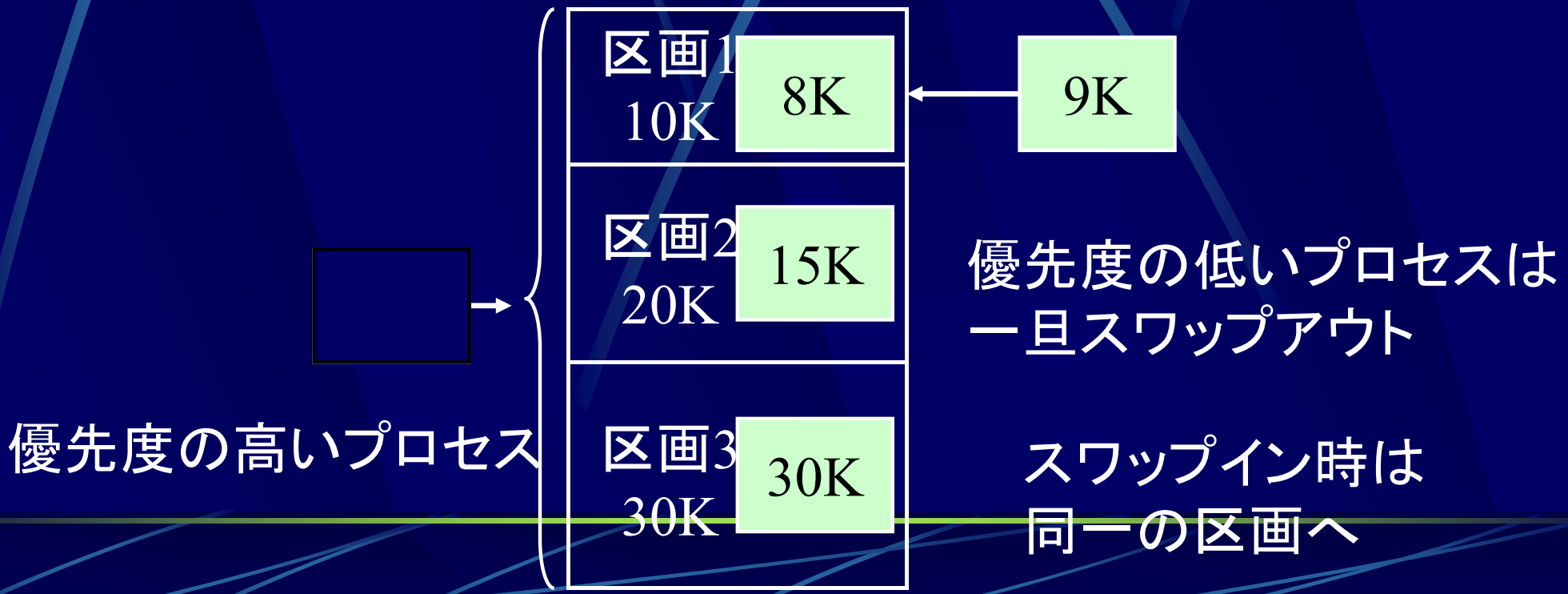
- 方法2-2：最小区画選択, 空き区画があるときはキューの実行順序を変更



区画2が空いているので
15Kのプロセスを
先に割り付け

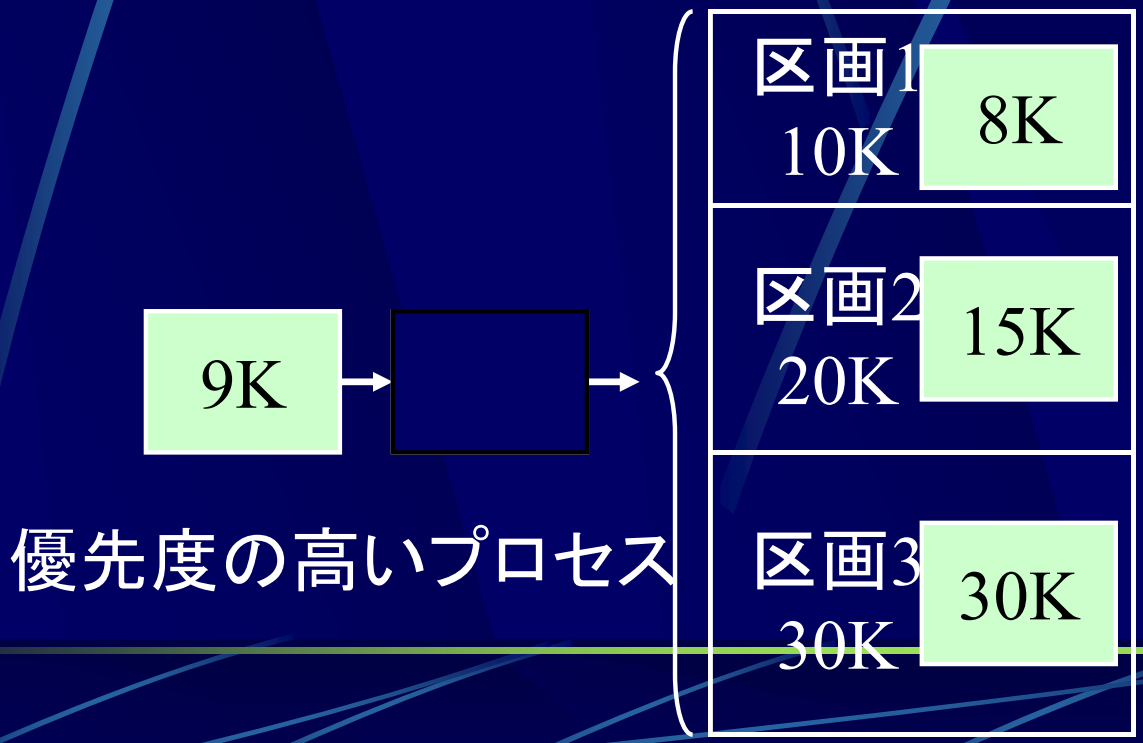
静的再配置とスワッピング

- 静的再配置とスワッピング
  - 現在実行中のプロセスよりも優先度が高いプロセスが来ればスワップアウト



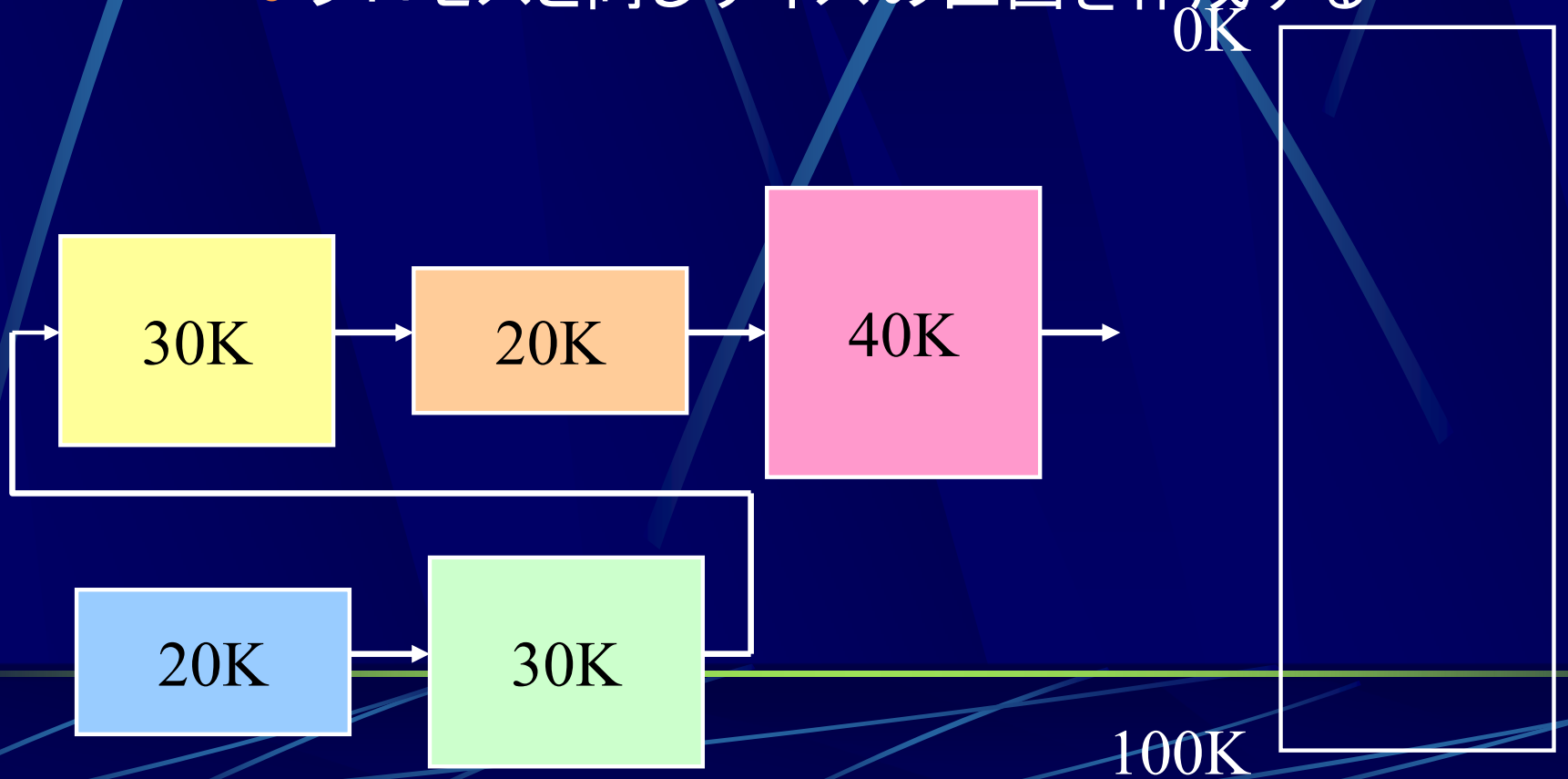
動的再配置とスワッピング

- 動的再配置とスワッピング
  - 現在実行中のプロセスよりも優先度が高いプロセスが来ればスワップアウト



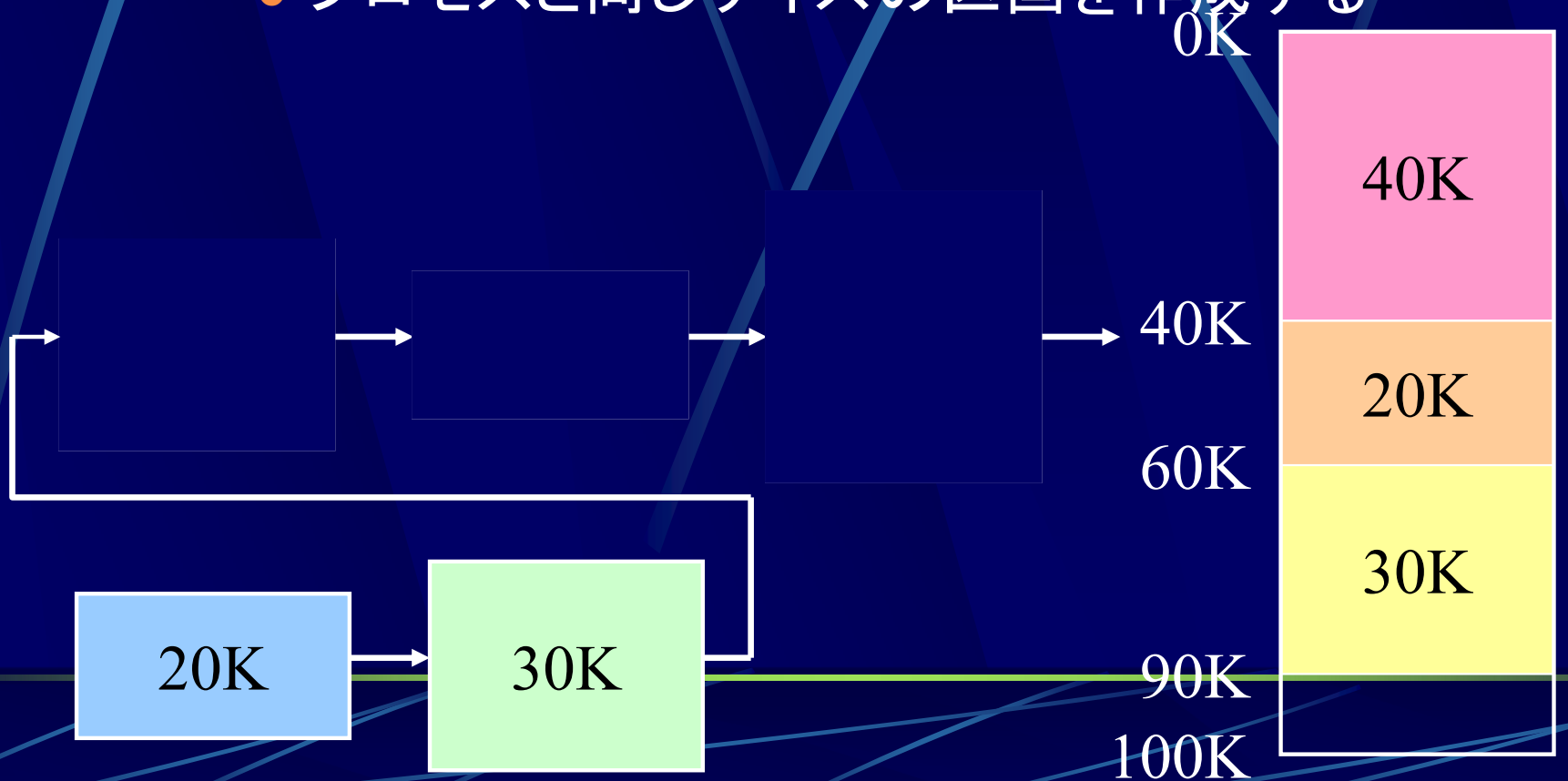
可変区画割り付け
(dynamic partition allocation)

- 可変区画割り付け
  - 区画のサイズを動的に変化させる
    - プロセスと同じサイズの区画を作成する



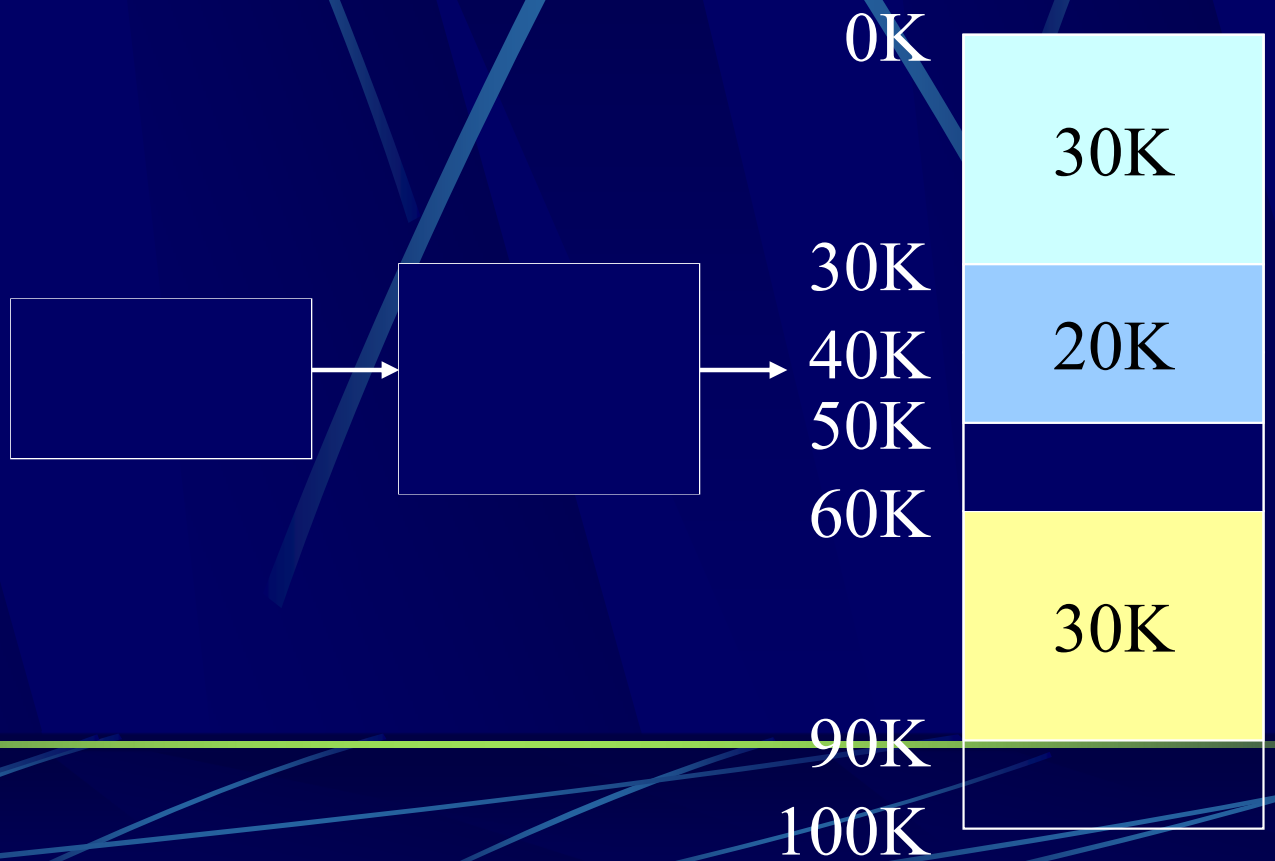
可変区画割り付け
(dynamic partition allocation)

- 可変区画割り付け
  - 区画のサイズを動的に変化させる
    - プロセスと同じサイズの区画を作成する



可変区画割り付け

- 可変区画割り付け
  - 区画のサイズを動的に変化させる
    - プロセスと同じサイズの区画を作成する

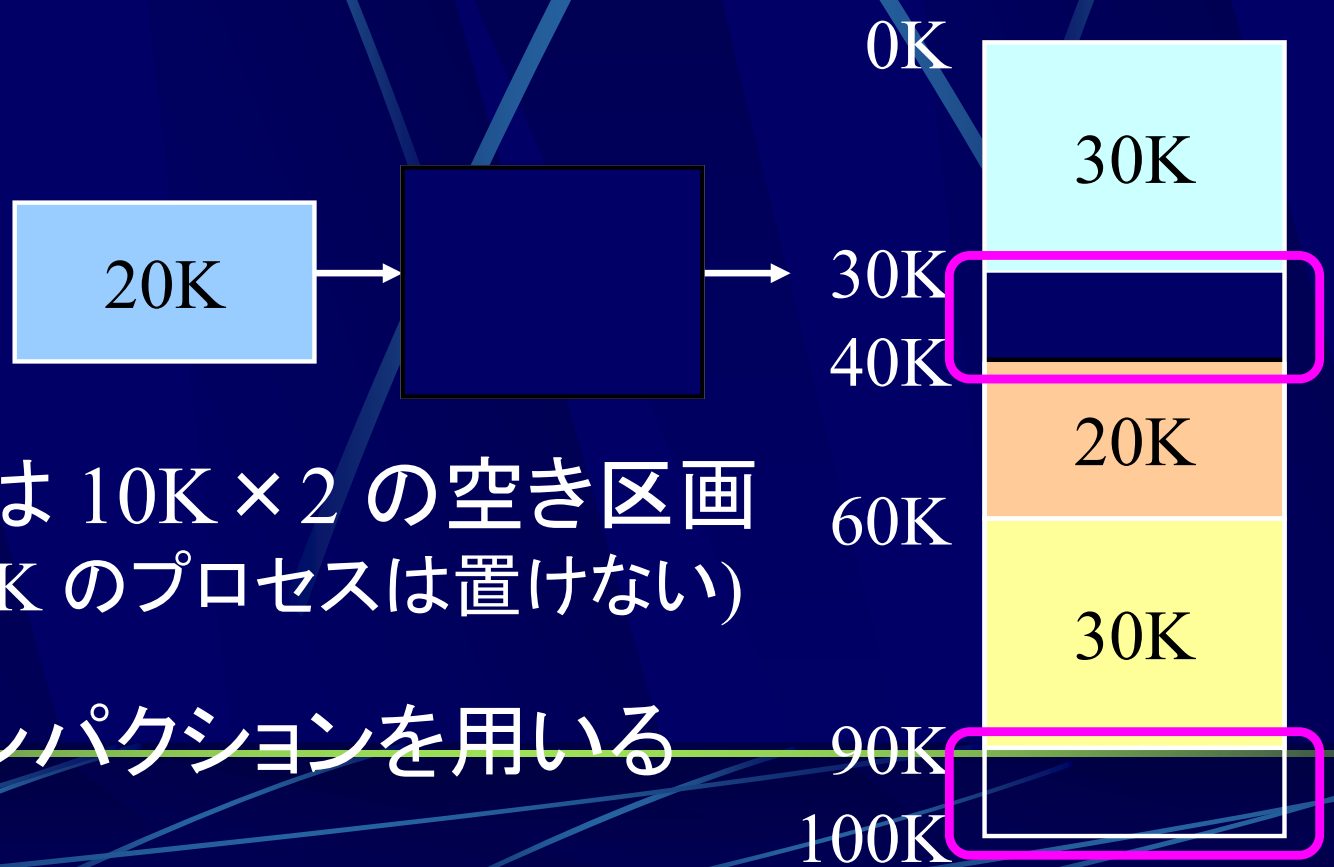


可変区画割り付けの長所と短所

- 可変区画割り付けの長所
  - 内部断片化が生じない
    - 固定区画割り付けよりもメモリ効率が上昇
- 可変区画割り付けの短所
  - 外部断片化が生じる
    - 処理の進行に伴い小さな空き領域が増加
  - 空き領域の検索コストがかかる
    - プロセスに割り当てる空き領域を探すコストが増大

可変区画割り付けの短所

- 外部断片化
  - 処理の進行に伴い小さな空き領域が増加
    (統計的には全体の50%が無駄になる)

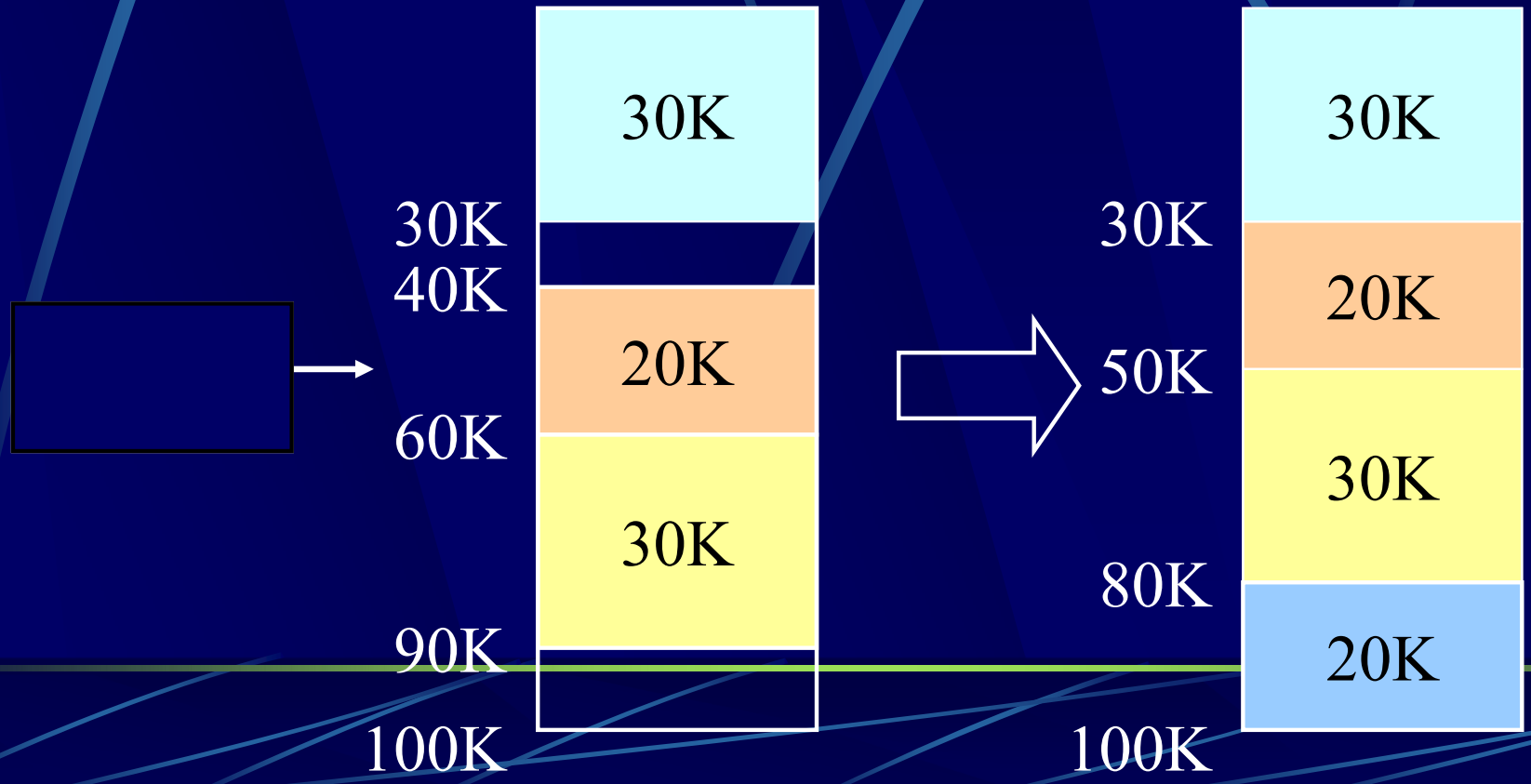


メモリには 10K×2 の空き区画
(しかし 20K のプロセスは置けない)

コンパクションを用いる

コンパクション(compaction)

- コンパクション(compaction)
  - 断片化している複数の空き領域をまとめて1つの大きな空き領域にする



コンパクションの長所と短所

- コンパクションの長所
  - 外部断片化を無くしメモリを有効利用できる
- コンパクションの短所
  - コンパクション中はプロセスの実行ができない

空き領域への割り付け

どの領域に割り付ける?

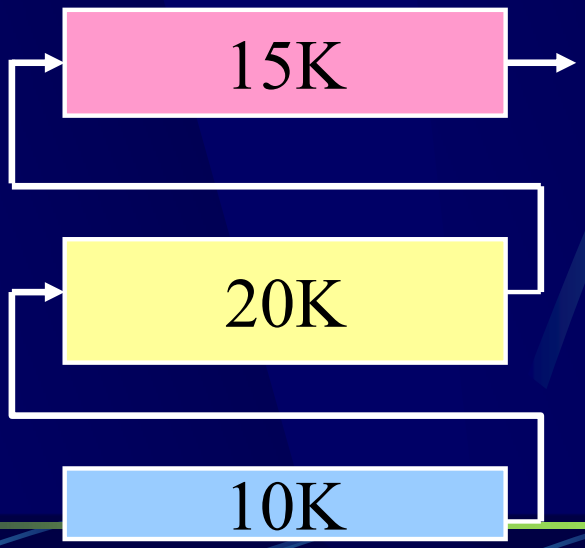
空き領域への割り付け

充分な大きさの空き領域が複数あるときにどこに割り付けするか?

- 先頭一致(first-fit)
  - 先頭の空き領域に割り付け
- 最良一致(best-fit)
  - 最も小さい空き領域に割り付け
- 最悪一致, 次一致(worst-fit, next-fit)
  - 最も大きい空き領域に割り付け

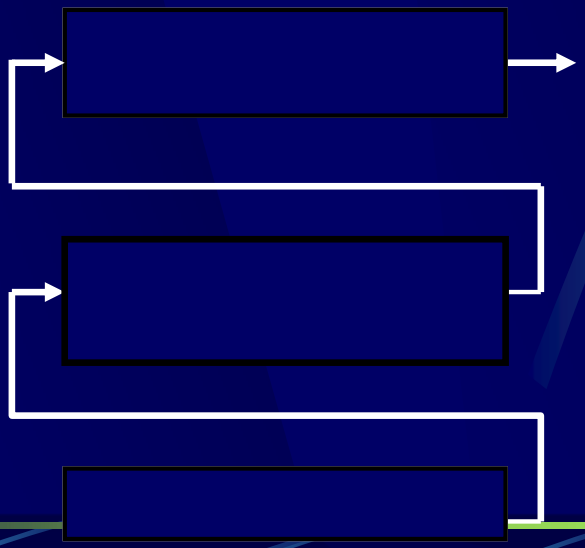
空き領域への割り付け

各プロセスを
どこに割り当てる？



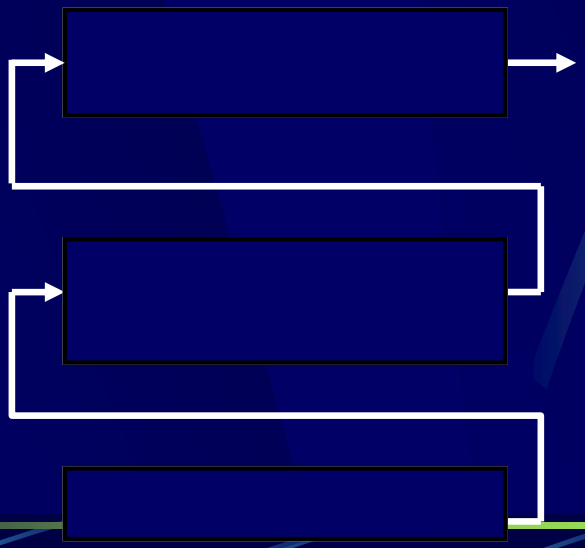
空き領域への割り付け

- 先頭一致(first-fit)
  - 先頭の空き領域に割り付け



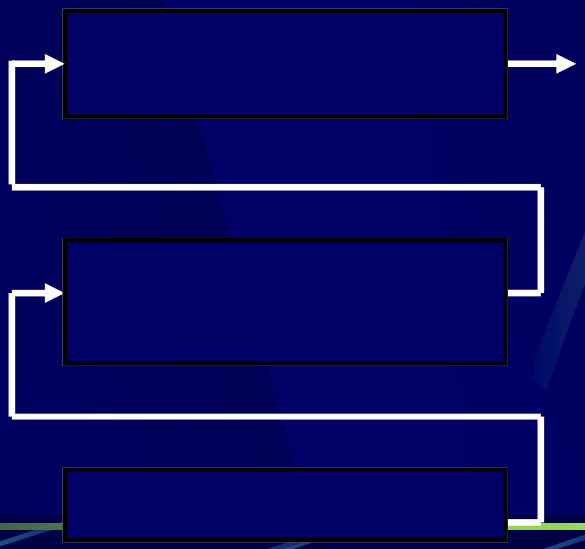
空き領域への割り付け

- 最良一致(best-fit)
  - 最小の空き領域に割り付け



空き領域への割り付け

- 最悪一致(worst-fit)
  - 最大の空き領域に割り付け



使用中
10K
使用中
20K
10K
使用中
20K
使用中
15K
10K
15K
使用中

空き領域への割り付け

割り付け法	長所
先頭一致	高速 アドレス上位に大きな空き領域ができ易い
最良一致	割り当て後にできる空き領域が小さい (外部断片化で無駄になる部分が小さい)
最悪一致	割り当て後にできる空き領域が大きい (空き領域に他のプロセスを割り当て可能)

空き領域の領域管理

- 空き領域の検索
  - プロセスは領域の割り付けと解放を繰り返す
    - 空き領域の数は常に増減

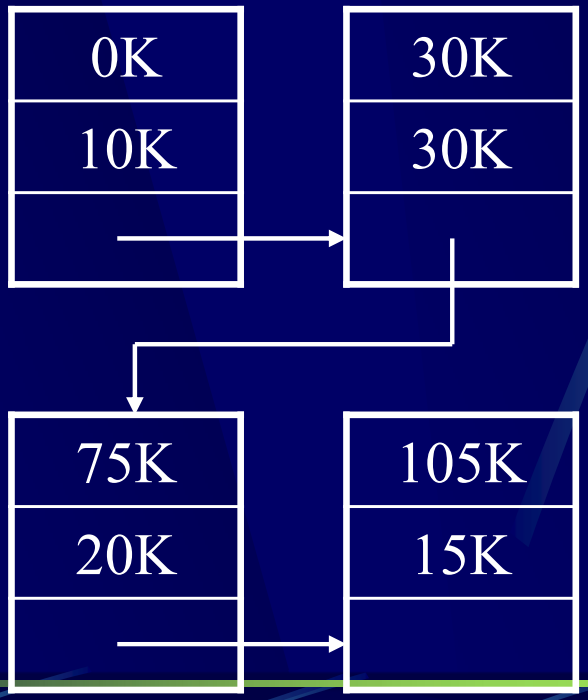
空き領域の高速な検索が必要

領域管理が必要

空き領域の領域管理

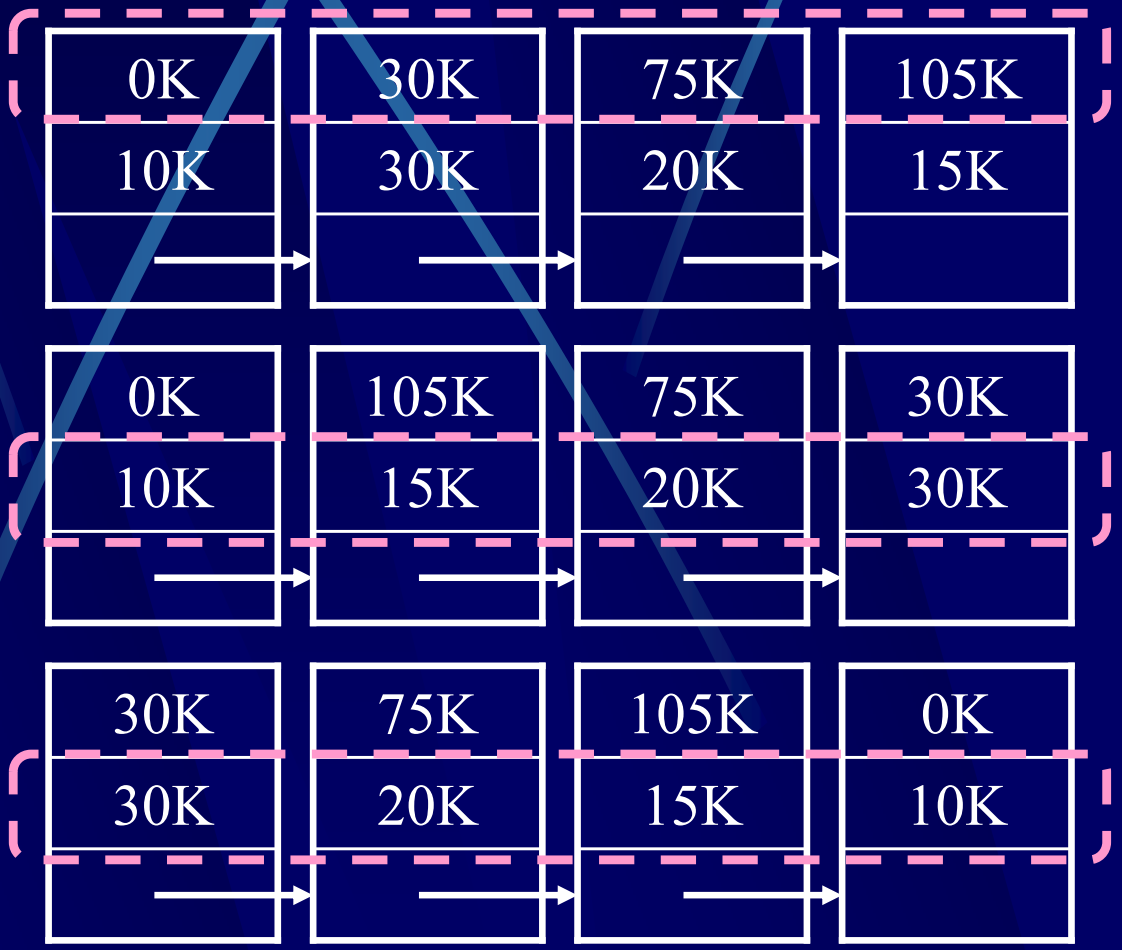
- 空き領域の領域管理方式
  - リスト方式(list)
    - 1つの空き領域を1エントリーとしてリストを作成
  - ビットマップ方式(bit-map)
    - 領域を一定サイズのブロックに分割、ブロック毎に空き/使用中 を管理

空き領域の領域管理
リスト方式(list)



空き領域の領域管理
リスト方式

- 先頭一致
  - アドレス順に
- 最良一致
  - サイズの昇順に
- 最悪一致
  - サイズの降順に



空き領域の領域管理
ビットマップ方式(bit-map)

0	0
5	0
10	1
15	1
20	1
25	1
30	0
35	0
40	0
45	0
50	0
55	0

60	1
65	1
70	1
75	0
80	0
85	0
90	0
95	1
100	1
105	0
110	0
115	0

空き領域の領域管理

管理方式	長所
リスト方式	空き領域の検索が高速 必用サイズの空き領域を見つけやすい
ビットマップ方式	特定の領域へのアクセスが高速 空き領域が増えてもアクセス時間は同じ

バディシステム(buddy system)

- バディシステム(buddy system)
  - プロセスが必要とするサイズ以上のサイズ 2^k $(L \leqq k \leqq U)$の区画を割り付け
    - 2^L : 区画の最小サイズ
    - 2^U : 区画の最大サイズ

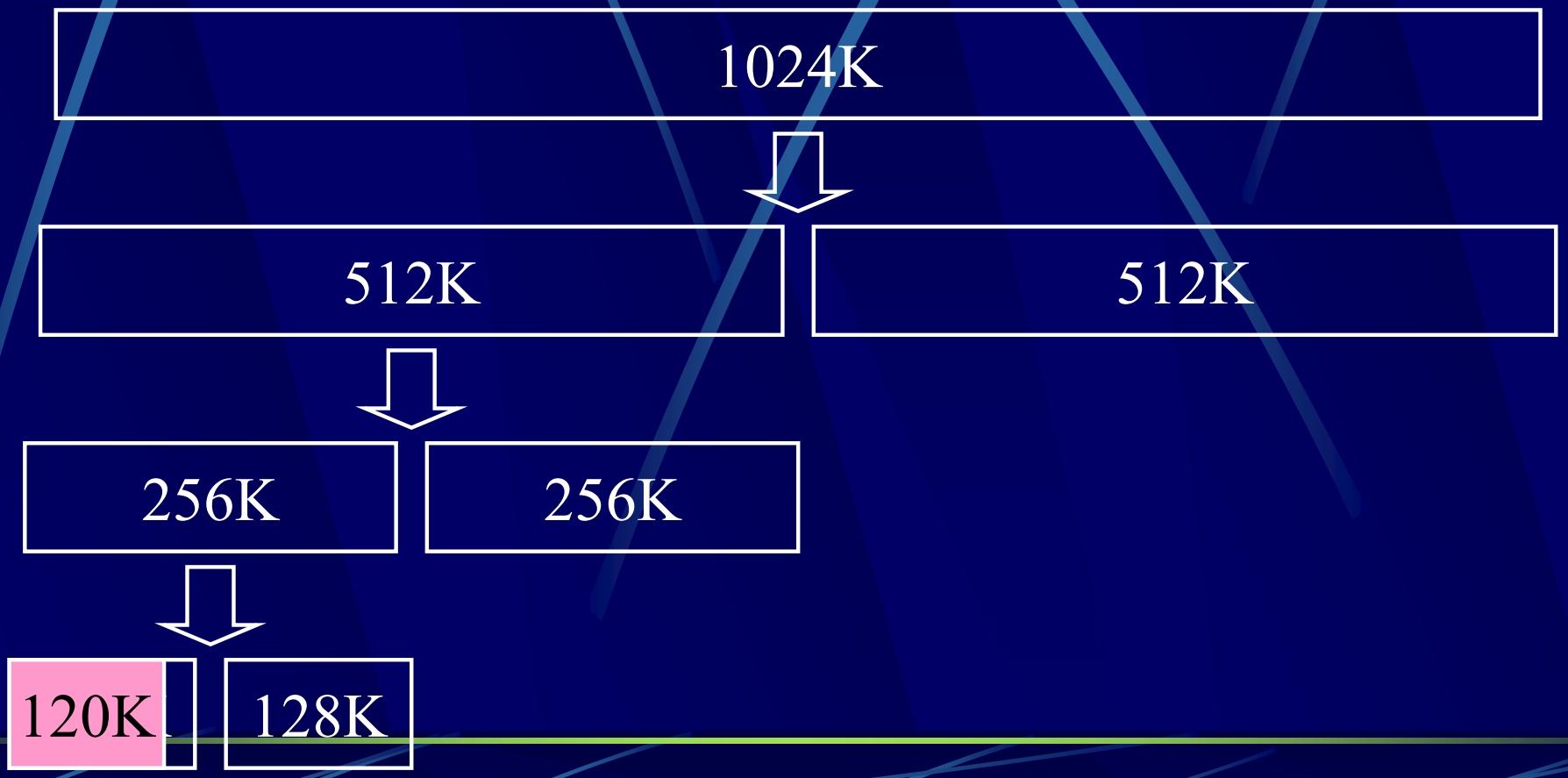
  初期状態ではサイズ 2^U の1つの区画とする

  プロセスが必要とするサイズに応じて
  区画を 1/2, 1/4, … に分割していく

バディシステム

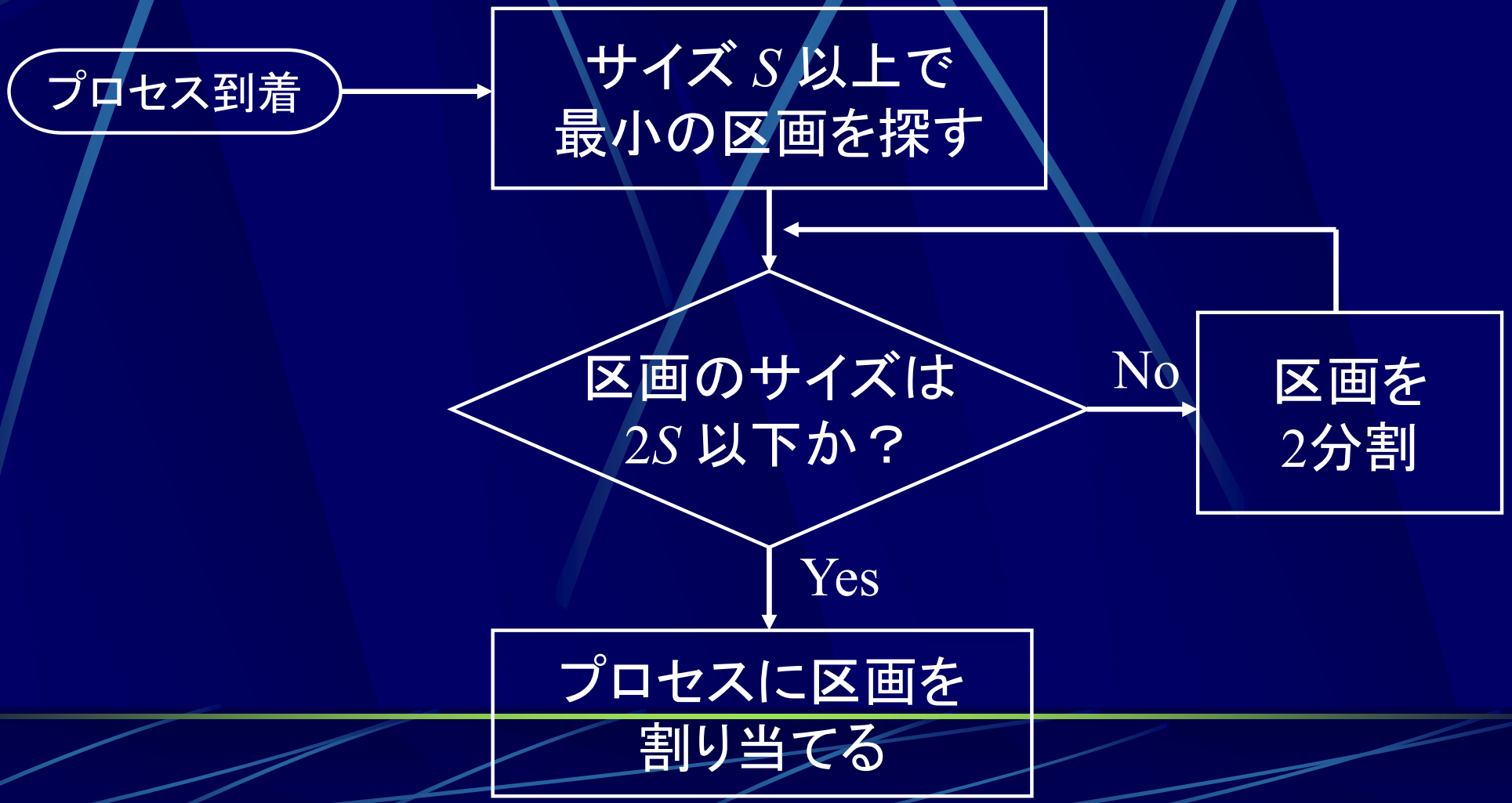
例：$2^U = 1024$K の場合

新規プロセス 120K

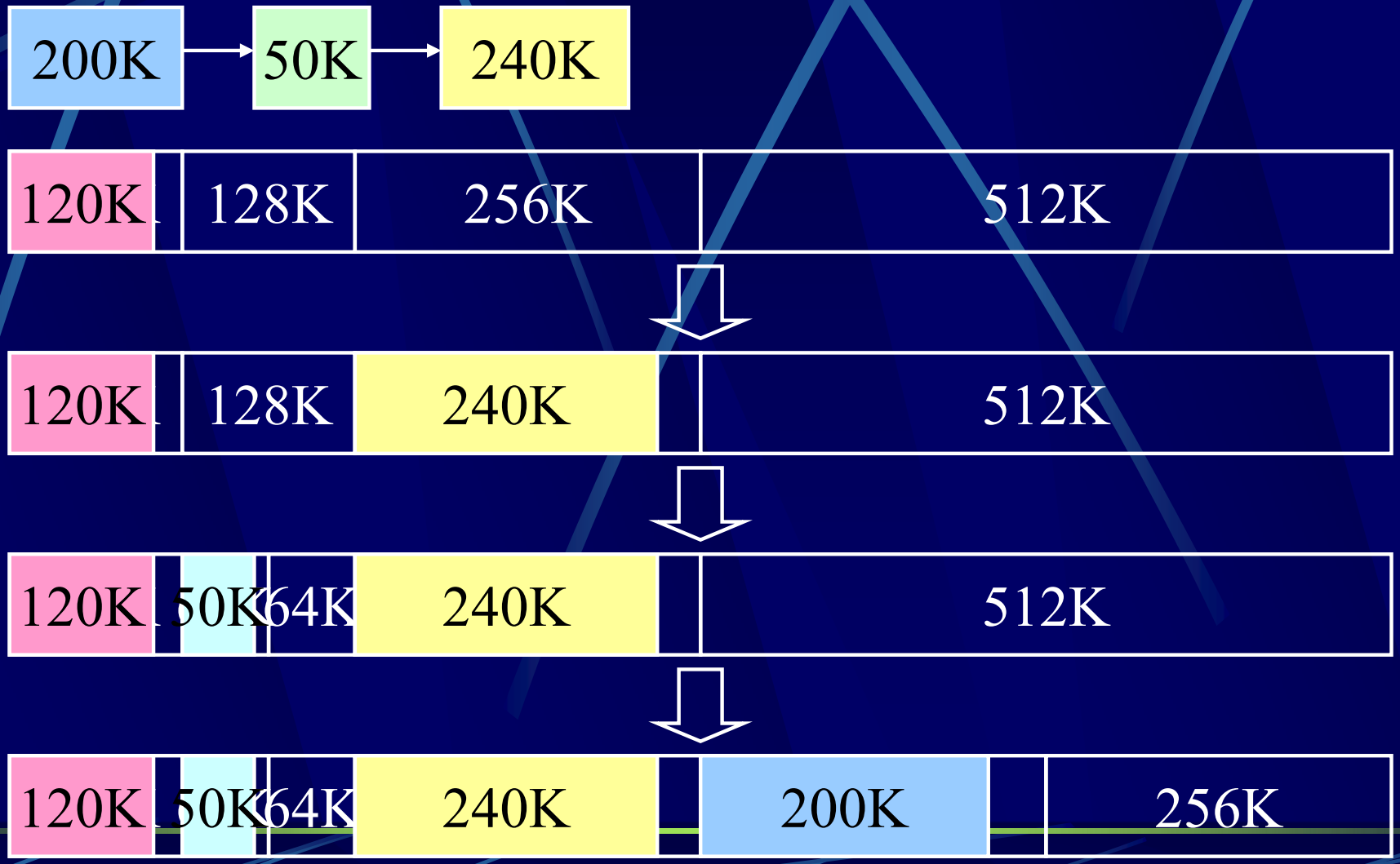


バディシステム

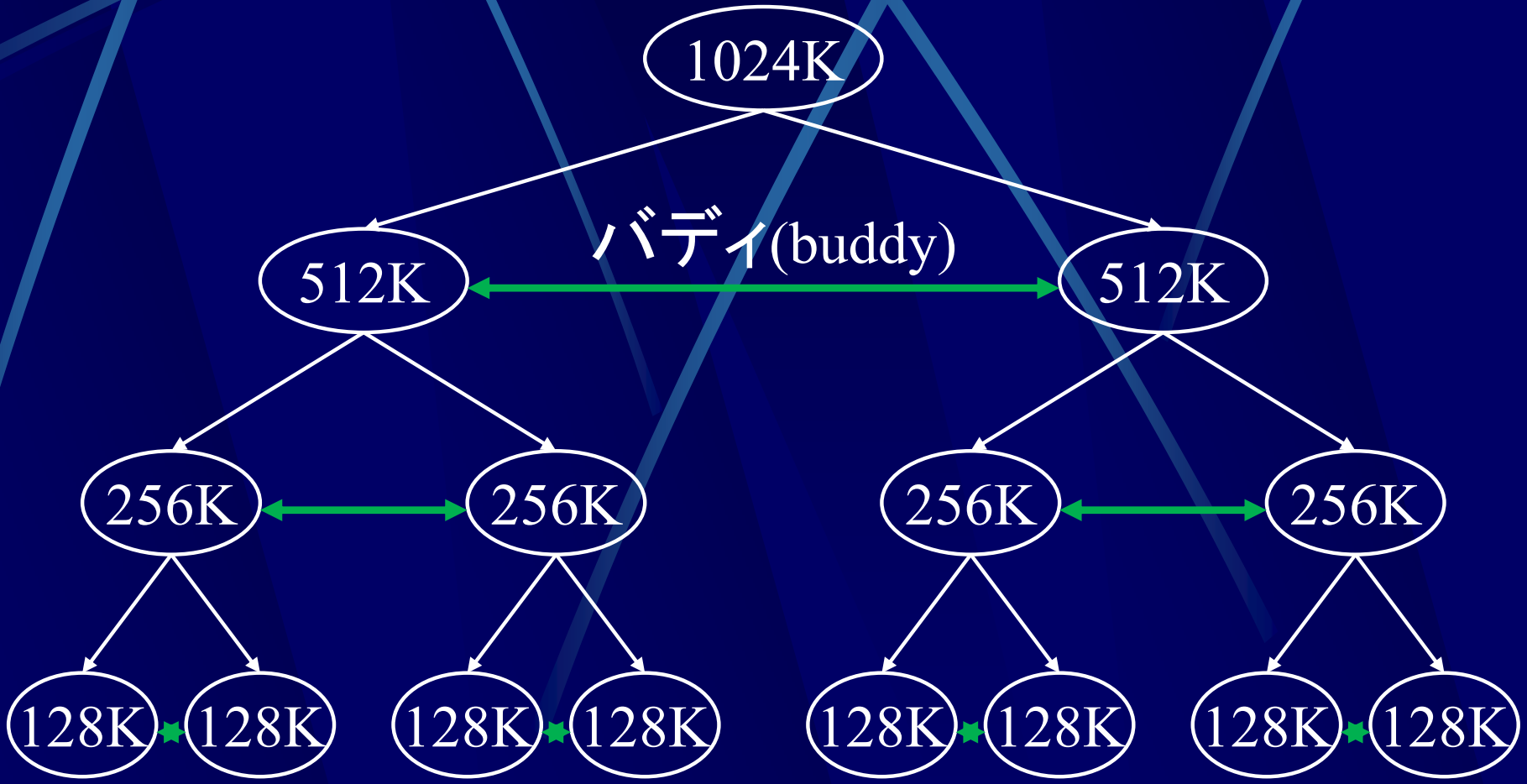
サイズ S のプロセスが到着した場合



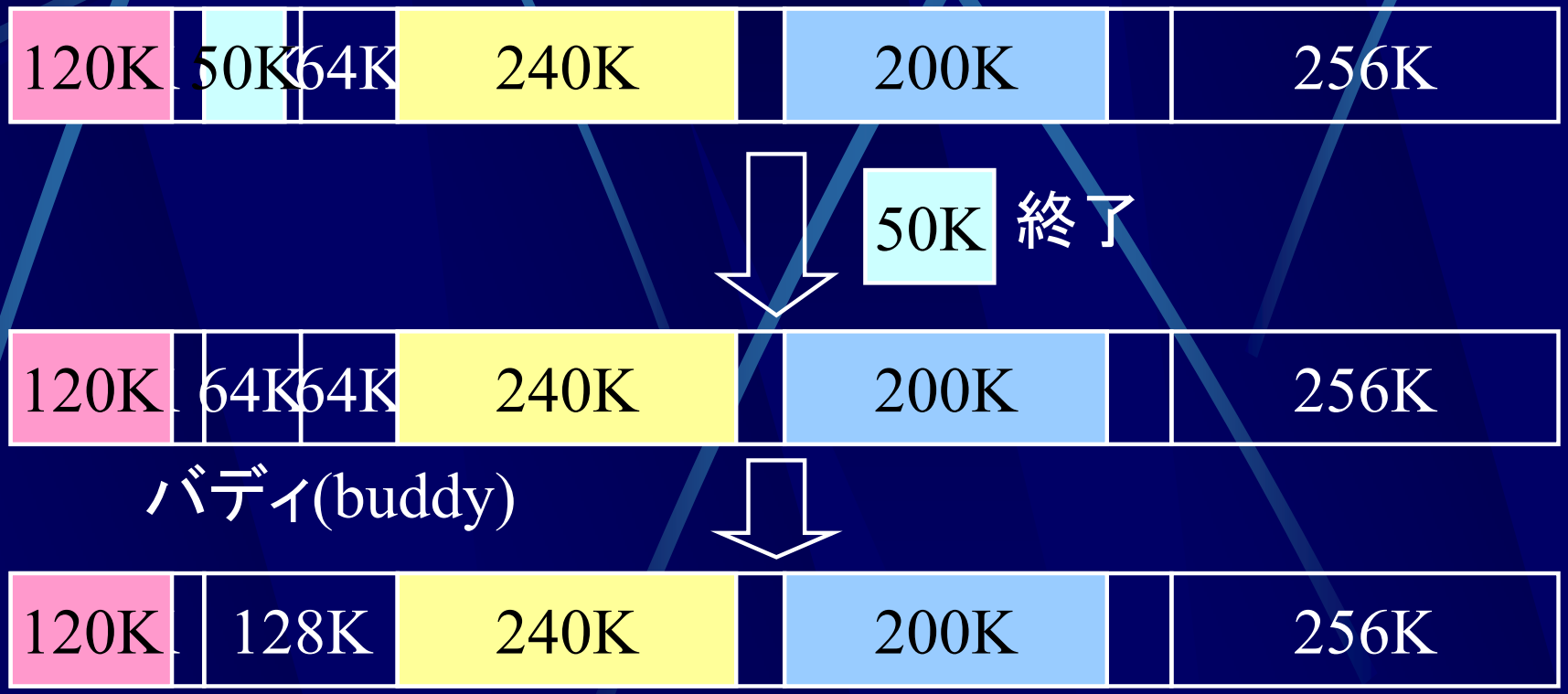
バディシステム


200K
50K
240K
120K
128K
256K
512K
120K
128K
240K
512K
120K
50K
64K
240K
512K
120K
50K
64K
240K
200K
256K

バディシステム

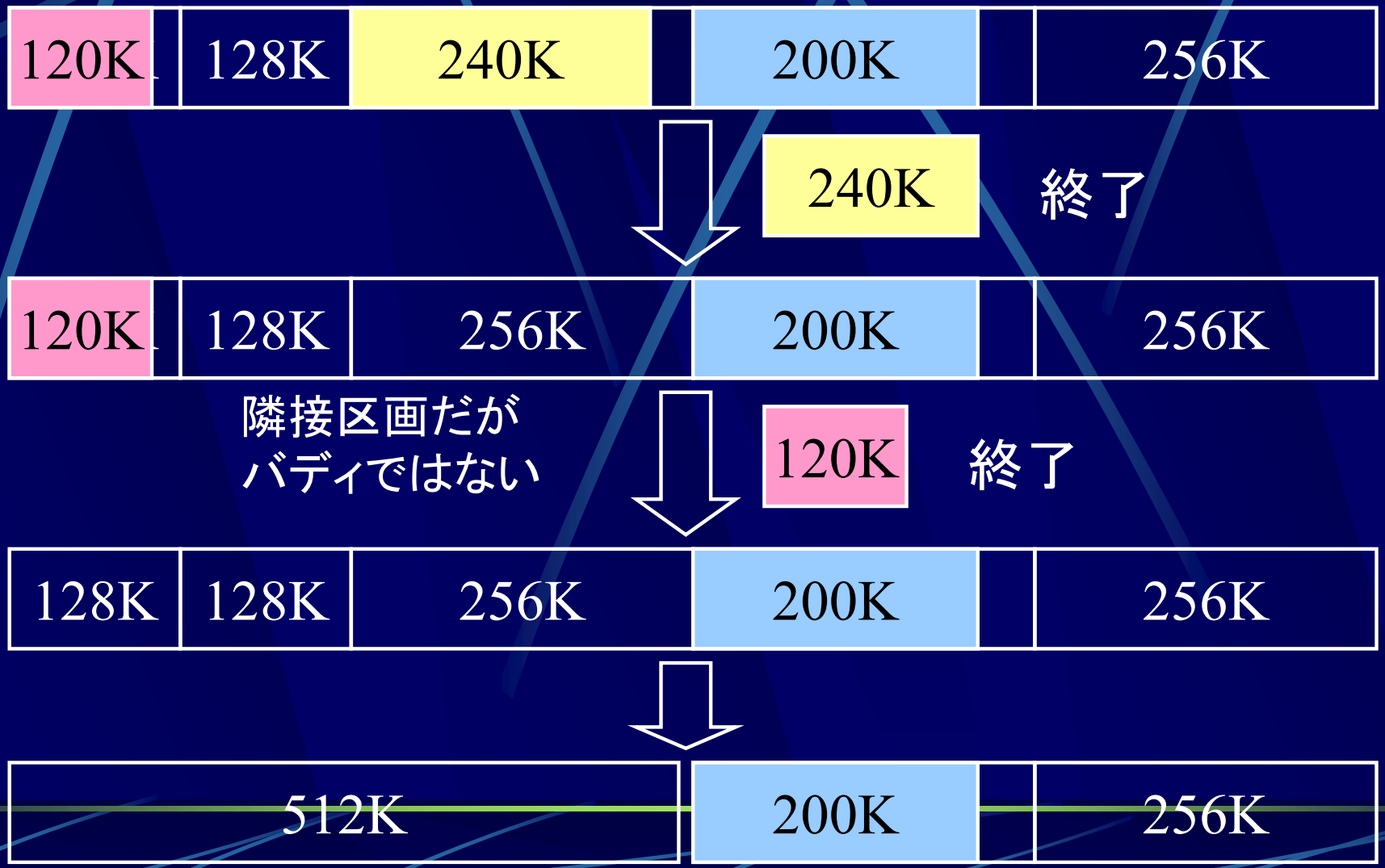

1024K
バディ(buddy)
512K
512K
256K
256K
256K
256K
128K
128K
128K
128K
128K
128K
128K
128K

バディシステム

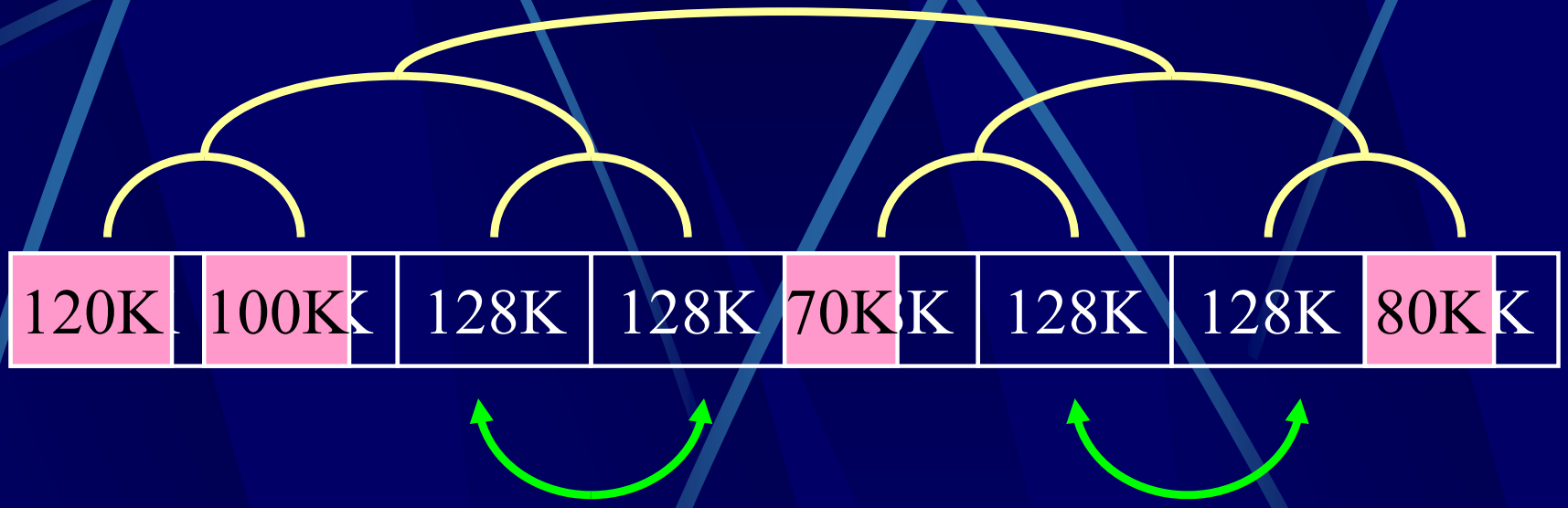


プロセス終了時にペアとなる隣接区画(buddy)が
空いていれば統合して2倍の大きさの空き領域とする

バディシステム


120K
128K
240K
200K
256K
240K
終了
120K
128K
256K
200K
256K
隣接区画だが
バディではない
120K
終了
128K
128K
256K
200K
256K
512K
200K
256K

バディシステム

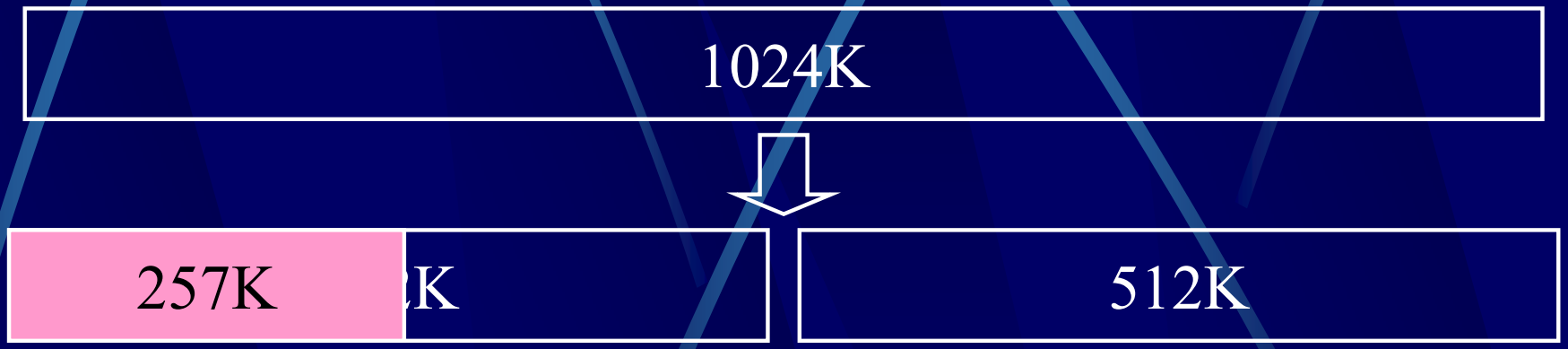


統合できる空き区画はバディのみ

バディシステム

例：$2^U = 1024$K の場合

新規プロセス 257K



512Kの区画に257Kのプロセス
＝約50%が内部断片化

内部断片化は最大で50%

記憶保護(memory protection)

OS, ユーザプログラム：
全て1つのメモリ上に置かれる

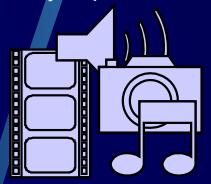
⇩

メモリのOS, 他のユーザプログラムの
領域を不当に書き換えてはならない

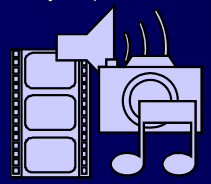
- 記憶保護(memory protection)
  - OS領域をユーザプログラムの不当アクセスから保護
  - ユーザプログラム間で不当なアクセスから互いに保護

記憶保護

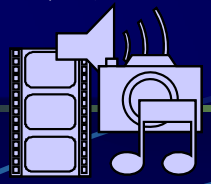
アプリケーション1

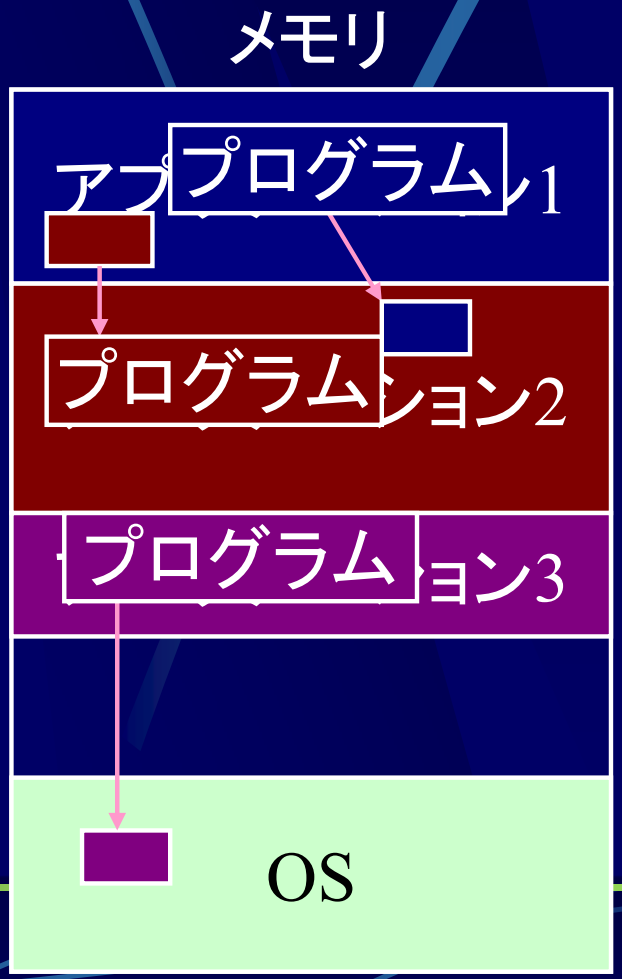


アプリケーション2



アプリケーション3





1が2の領域へ
不当な書き込み

2が1の領域から
不当な読み込み

3がOSの領域に
不当な書き込み

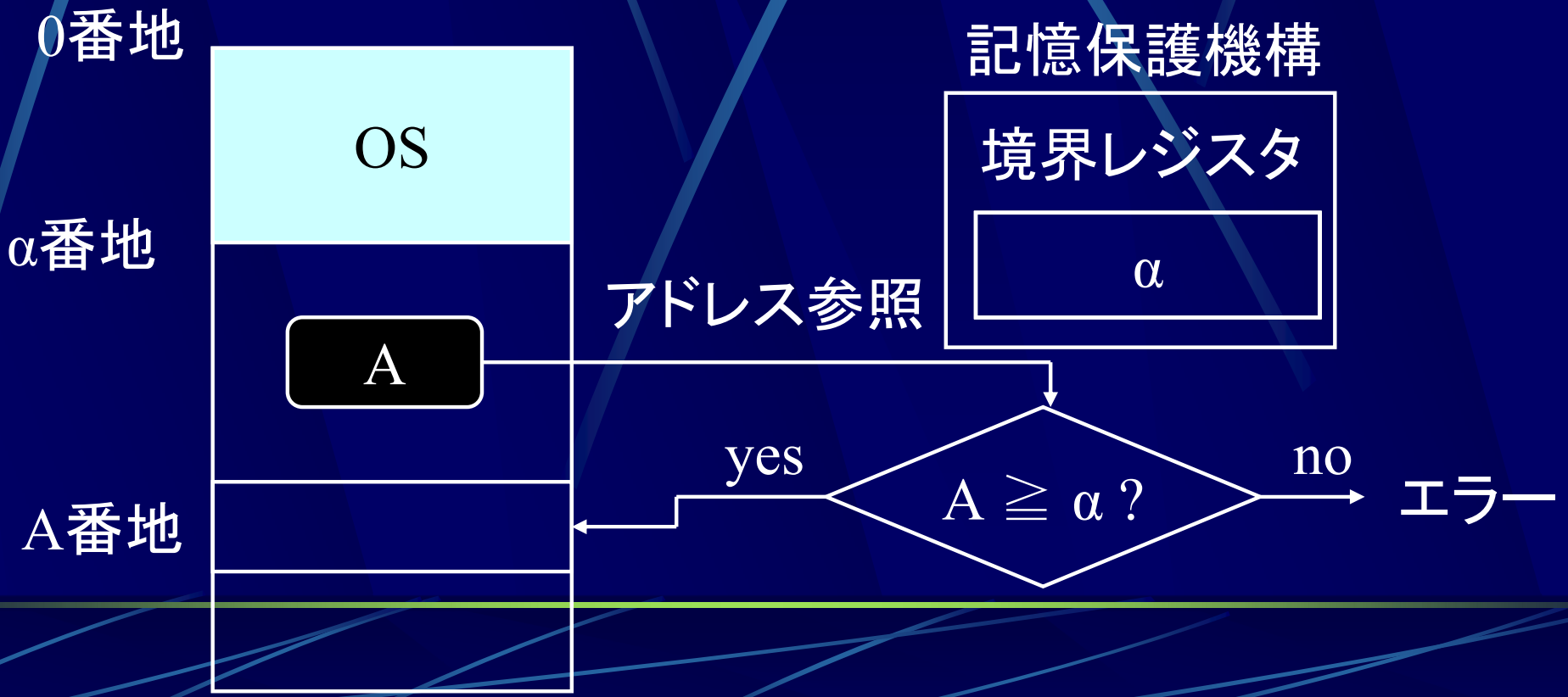
アプリケーションを
停止する

記憶保護

- 単一ユーザの記憶保護
  - OS領域とユーザ領域の境界を管理
- マルチプログラミングの記憶保護
  - 各プロセス領域の下限と上限を管理

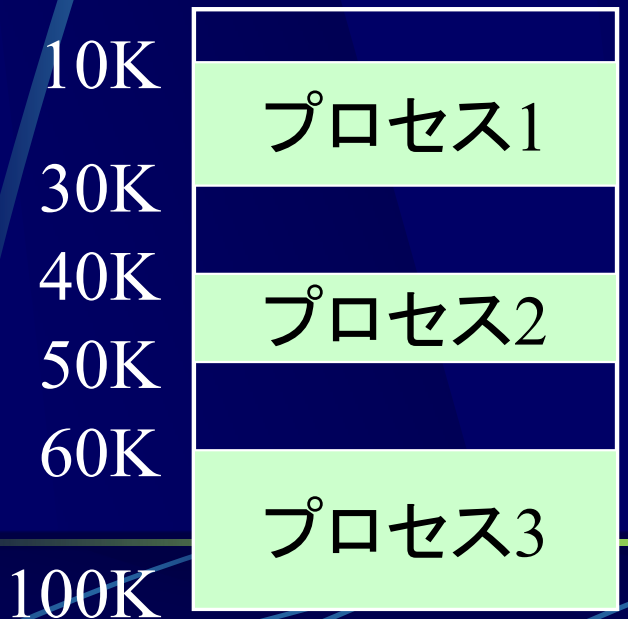
単一ユーザの記憶保護

- 境界レジスタ(boundary register)
  - OS領域とユーザ領域の境界アドレスに設定



マルチプログラムの記憶保護

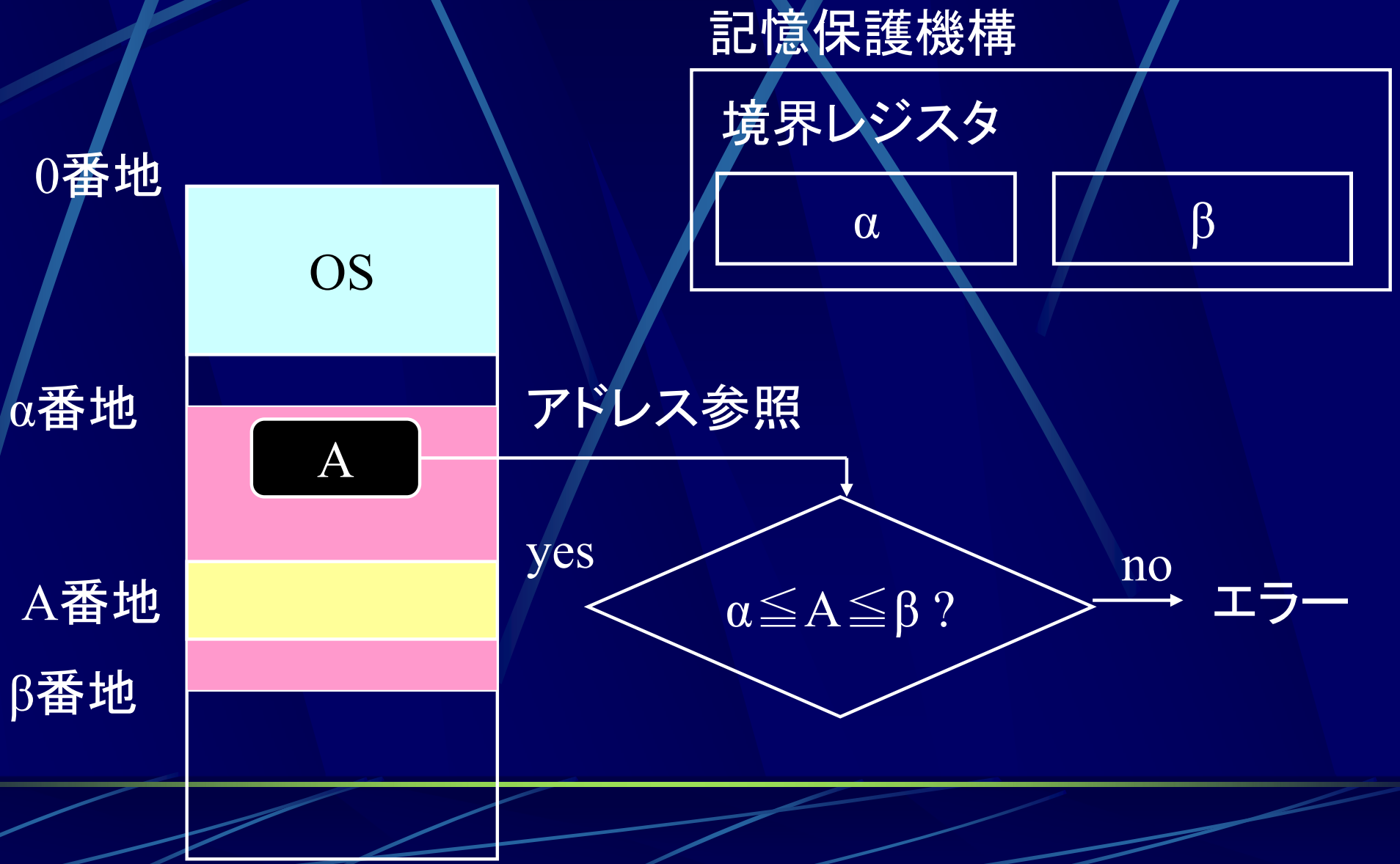
- マルチプログラムの記憶保護
  - プロセスごとに以下のどちらかの情報を管理
    - 先頭のアドレスと末尾のアドレス
    - 先頭のアドレスとプロセスサイズ



	先頭-末尾	先頭-サイズ
プロセス1：	10K-30K	10K-20K
プロセス2：	40K-50K	40K-10K
プロセス3：	60K-100K	60K-40K

2個の境界レジスタで管理

マルチプログラムの記憶保護



まとめ(単一連続割り付け)

- 再配置(relocation)
  - 相対番地で記述されたプログラムを配置
- スワッピング(swapping)
  - 待ち状態のプログラムを2次記憶に退避
- オーバレイ(overlay)
  - 必要な部分のみを主記憶上に読み込む

まとめ (区画割り付け)

- 固定区画割り付け(static partition allocation)
  - 区画の大きさは予め決定, プロセスが必要とするサイズ以上の区画に割り付け
- 可変区画割り付け(dynamic partition allocation)
  - 区画の大きさをプロセスに応じて変更
- バディシステム(buddy system)
  - プロセスが必要とするサイズ以上のサイズ 2^k の区画を割り付け

まとめ (区画割り付け)

割り付け法	長所	短所
固定区画 割り付け	空き領域の管理が容易 外部断片化が起きにくい	内部断片化
可変区画 割り付け	内部断片化が起きない	外部断片化 空き領域の管理が難しい
バディ システム	上2つのハイブリッド	内部断片化(最大50%)

まとめ(固定区画割り付け)

- 絶対番地式
  - 割り付け位置は固定
- 相対番地式
  - 再配置で割り付け
  - 静的再配置とFCFSスケジューリング
    - 最小区画選択
    - 最小空き区画選択
  - 静的再配置とスワッピング
  - 動的再配置とスワッピング

まとめ (可変区画割り付け)

- 先頭一致(first-fit)
  - 先頭の空き領域に割り付け
- 最良一致(best-fit)
  - 最も小さい空き領域に割り付け
- 最悪一致, 次一致(worst-fit, next-fit)
  - 最も大きい空き領域に割り付け