

複数OS走行環境における マルチコアCPUの 計算資源分割制御方法に関する研究

岡山大学 大学院 自然科学研究科
牛尾 裕

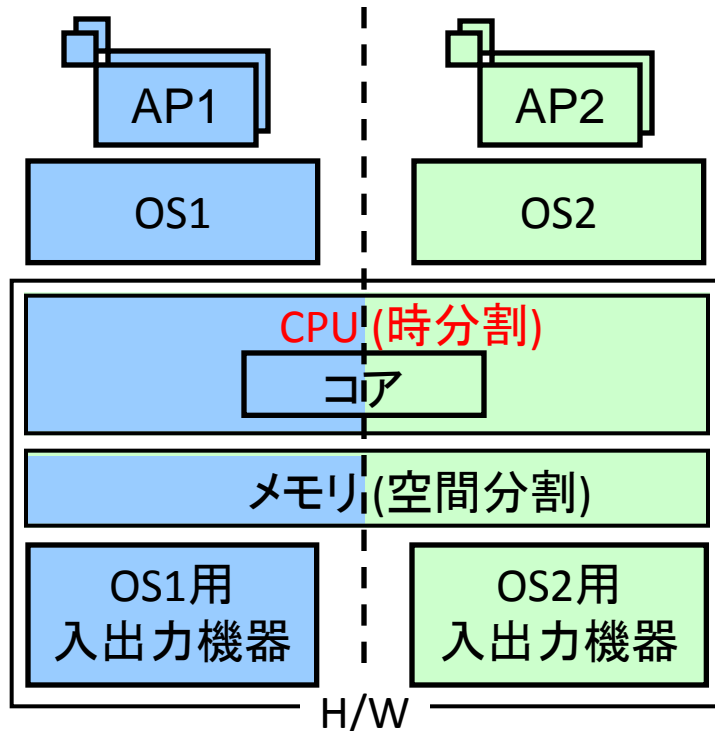
背景

マルチコアCPUを効率よく利用する方法

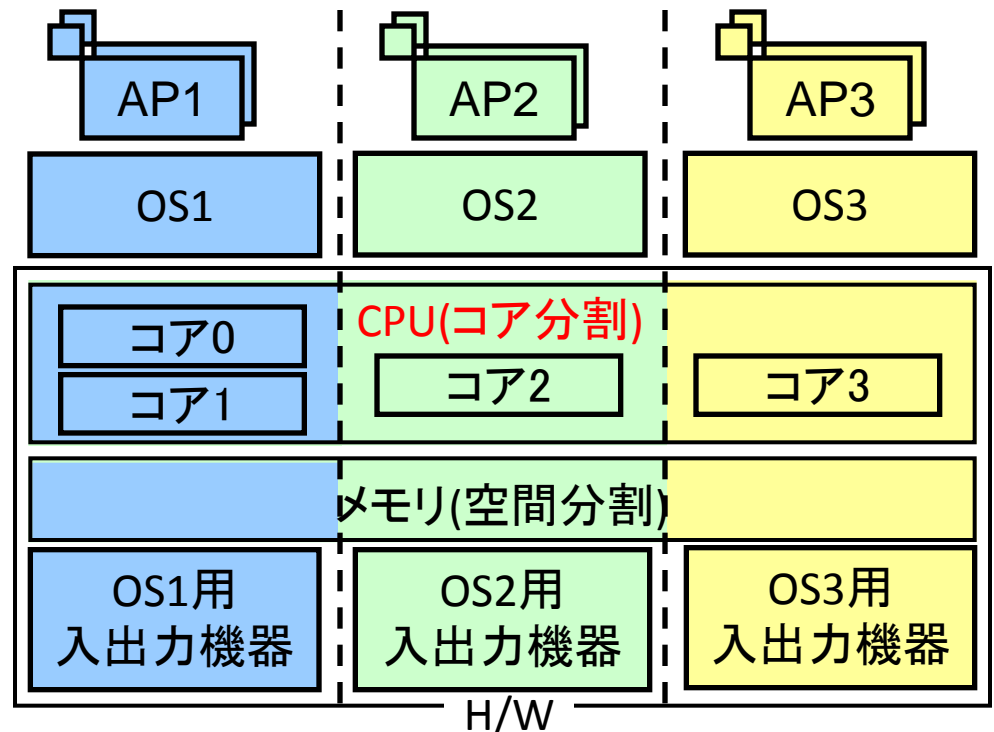
➡ 1台の計算機上で複数のOSを走行させる技術の研究が活発

TwinOSとMintオペレーティングシステム(Mint)が実現されている

(1) TwinOS



(2) Mint



CPU分割制御方法

<TwinOS>

シングルコアCPUを時分割制御

(利点) 並行に走行可能なOS数がコア数に依存しない

(欠点1) マルチコアCPUに対応していない

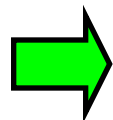
(欠点2) 計算資源の共有により独立性が低い

<Mint>

マルチコアCPUをコア分割制御

(利点) 計算資源を共有しないため各OSの独立性が高い

(欠点) コア数より多くのOSを並行に走行できない



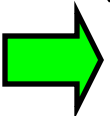
資源の分割粒度はコア数に依存するため粗い

目的

<現在の計算機>

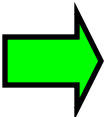
計算性能は入出力性能に比べ高い

(例) HDDのシーク時間によるCPU処理の遅延

 過剰に計算性能が割り当てられている

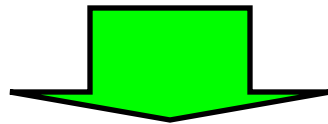
<Mintにおける計算資源分割>

マルチコアCPUをコア単位で分割

 分割粒度はマルチコアCPUのコア数に依存

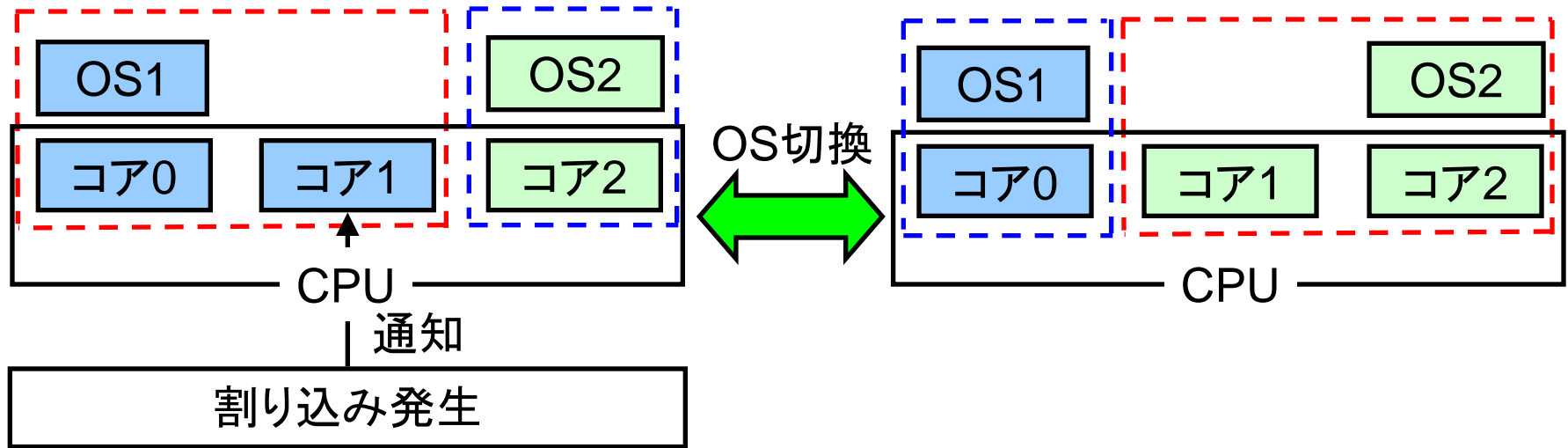
(要求) MintのCPUの分割粒度を細かくしたい

(例) 各OSに1.5コアの計算資源を割り当てる



コア分割制御に代わる計算資源分割制御方法を検討

併用分割制御法



コア分割制御と時分割制御を組み合わせた併用分割制御法を提案

<特徴>

(1) 並行に走行可能なOS数の制限なし

➡ 細かい計算資源の分割が可能

(2) 時分割制御しているコアは、走行OSを切り換える(OS切換)

➡ コア分割制御と比べ、各OSの独立性が低下

シングルコアの時分割制御の実装が必要

課題と設計

(課題1) メモリの分割と占有

- (1) 走行させるOSの数に合わせメモリを空間分割

(課題2) 各OSの起動

- (1) 最初に起動するOS(先行OS)の走行環境の保存
- (2) 先行OSから起動するOS(後続OS)のメモリ配置と起動

(課題3) 割り込みのフック

- (1) OS切換の契機に対応したフック方法

(課題4) OS切換

- (1) OSの走行環境の保存と復元
- (2) マッピングテーブル(MT)の変更

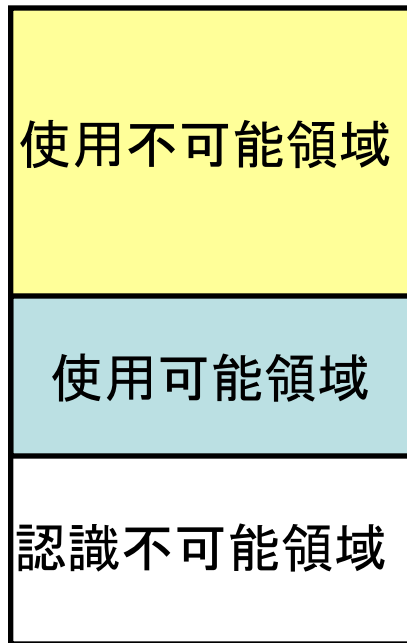
メモリの分割と占有

走行させるOSの数に合わせメモリを空間分割

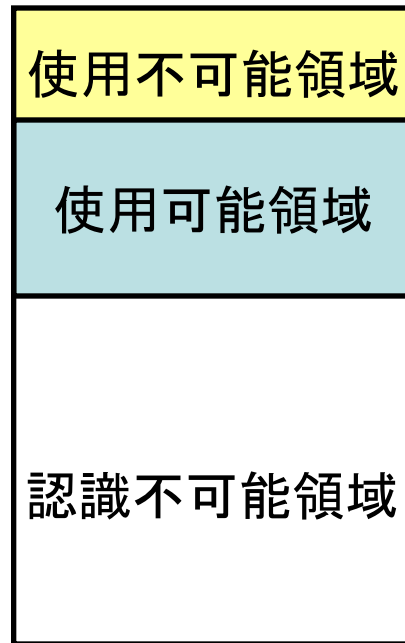
<先行OS>

BIOSコールにより取得したメモリマップを基にメモリ領域を認識

➡ メモリマップ取得直後にメモリマップを書き換える処理を追加



先行OSの
メモリマップ



後続OSの
メモリマップ

<後続OS>

先行OSが指定したメモリマップ
を基に起動

各OSの起動

(1) 複数のOSの同時起動は初期化処理の競合により困難

 OSを順次起動

(2) 後続OS起動時に先行OSの走行環境を保護する必要

 ソースコードを改変

<先行OSの改変>

(1) 後続OS起動前に4種類の情報を先行OSのデータ領域に保存

(A) OSの仮想空間情報

(B) 割り込み管理情報

(C) レジスタ情報

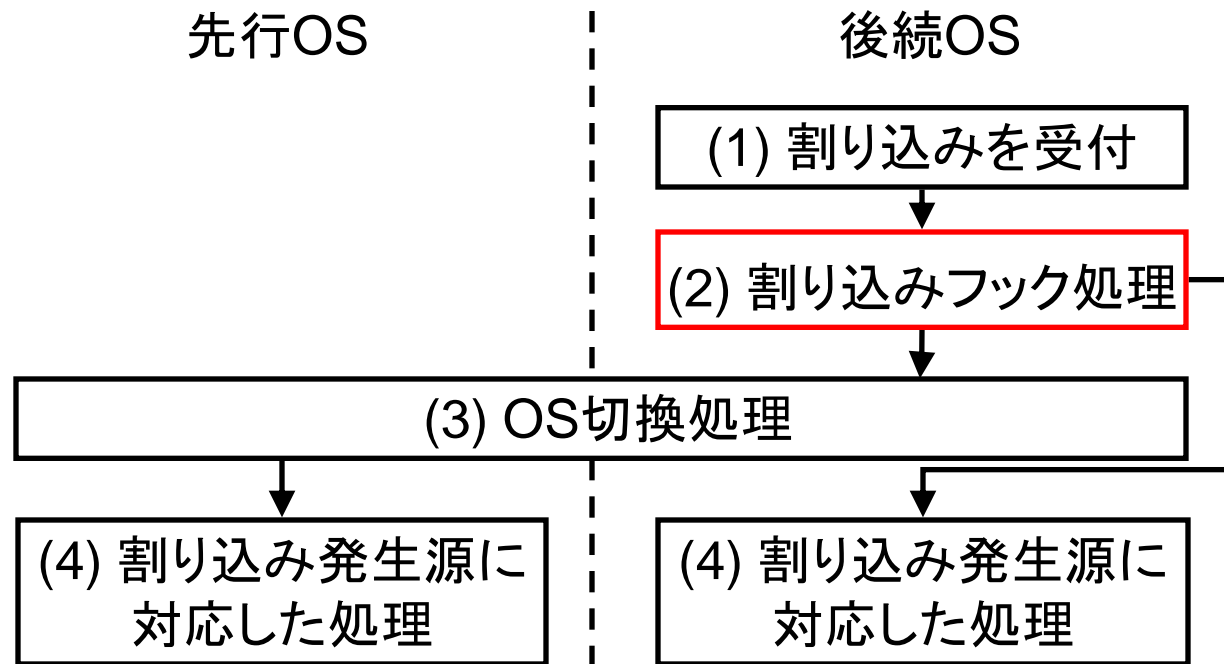
(D) 走行環境復元処理の開始アドレス情報

(2) 先行OSから指定したメモリマップを用いて後続OSを起動

<後続OSの改変>

先行OSと共有する機器は初期化処理を省略

割り込みのフック



割り込み発生源の違いにより動作が異なる

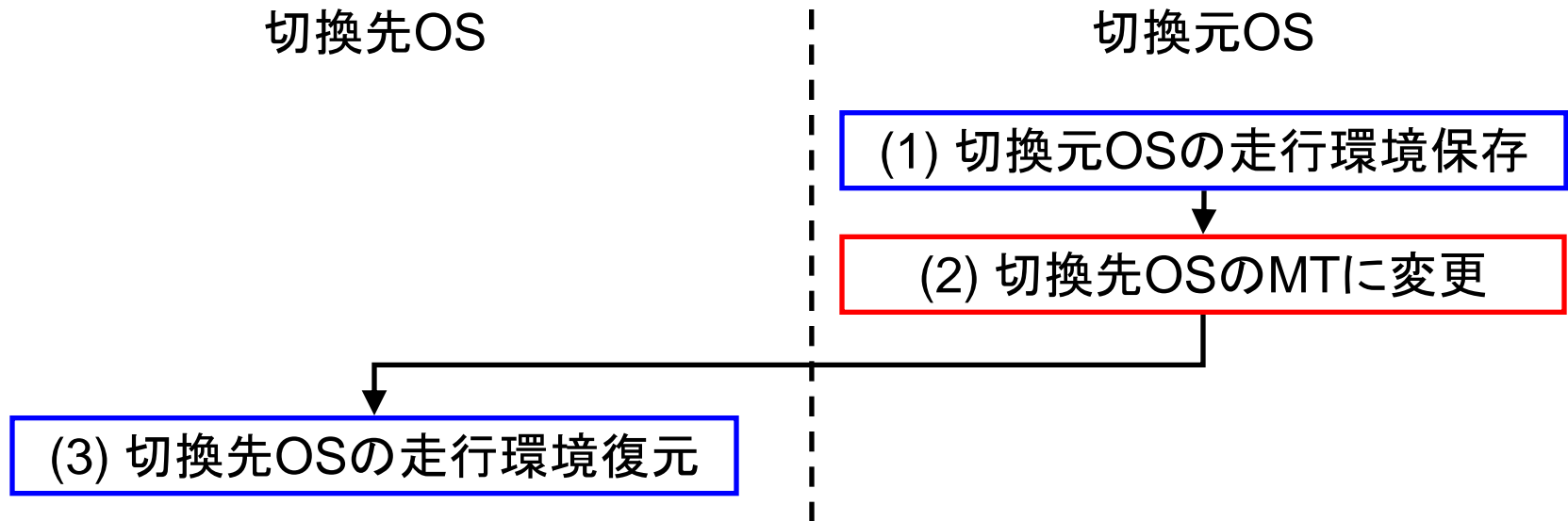
(1) ローカルタイマ割り込み

➡ OS切換を実行

(2) 入出力機器からの割り込み

➡ 割り込み識別番号を基に**走行OSの非占有機器からの割り込みの場合のみOS切換**

OS切換



(1)(3) OSの走行環境保存と復元

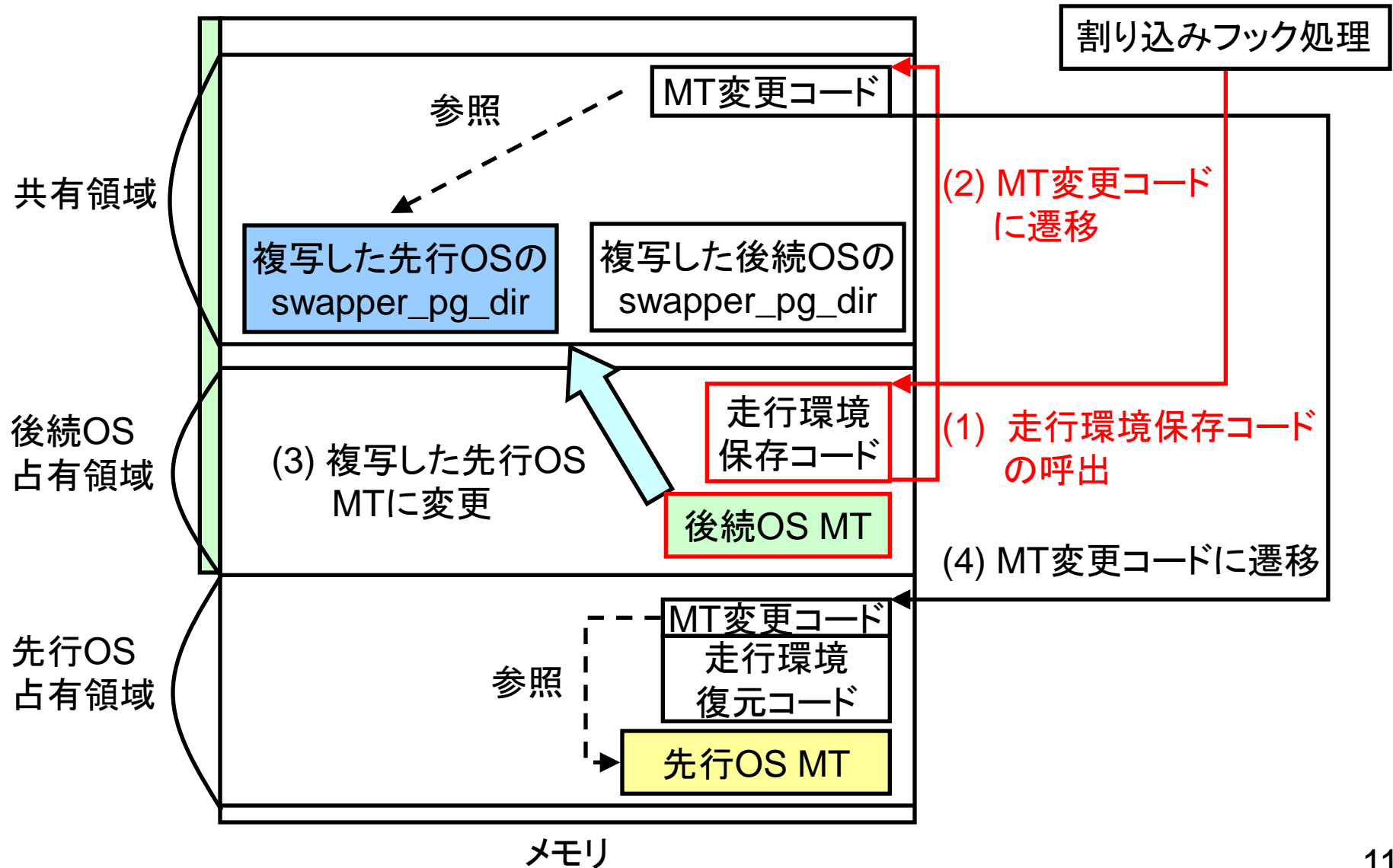
➡ 各OSの起動時の走行環境保護と同様

(2) 切換先OSのMTに変更

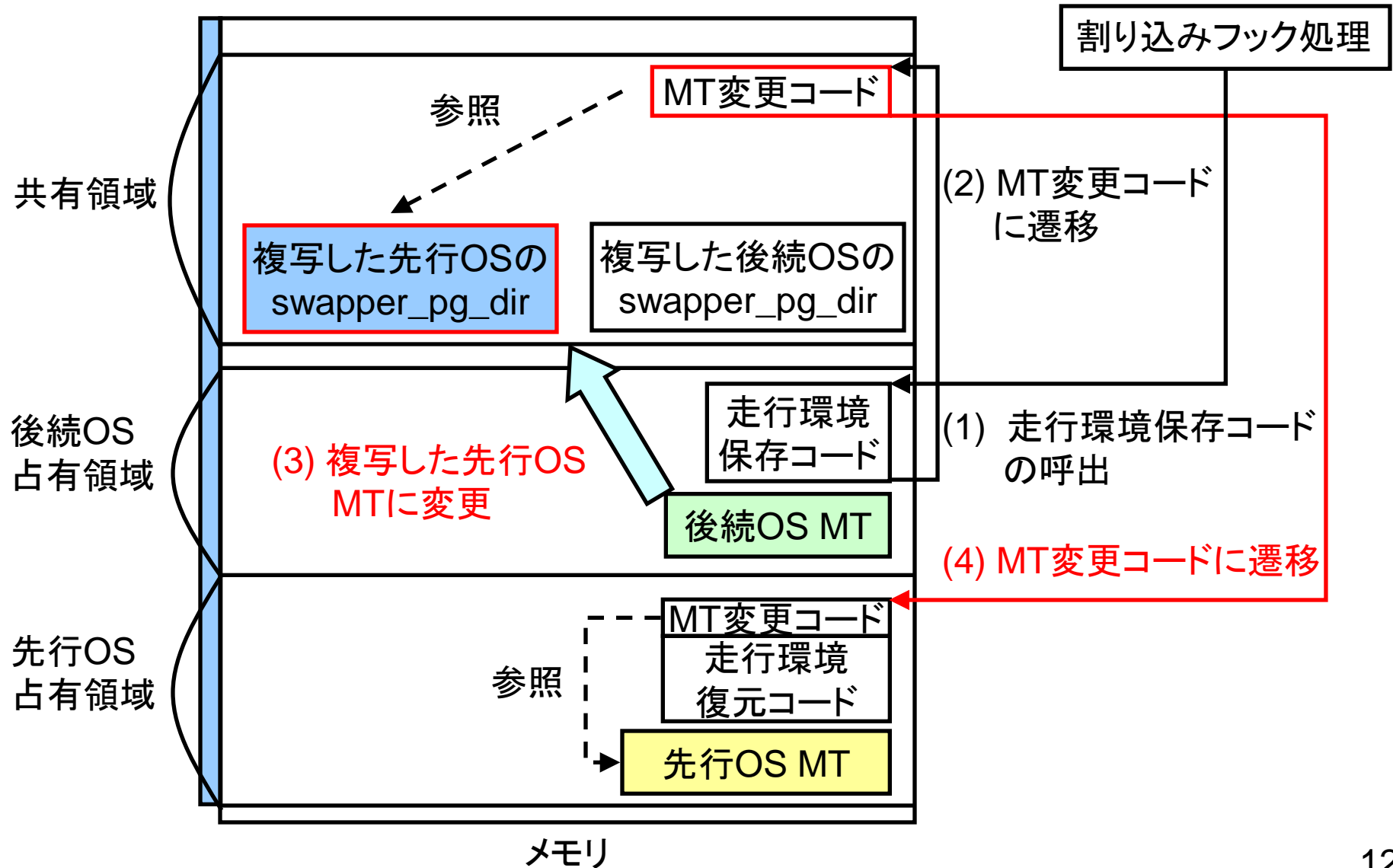
(2)終了アドレスと(3)開始アドレスを一致させる必要

➡ MTの配置場所を検討

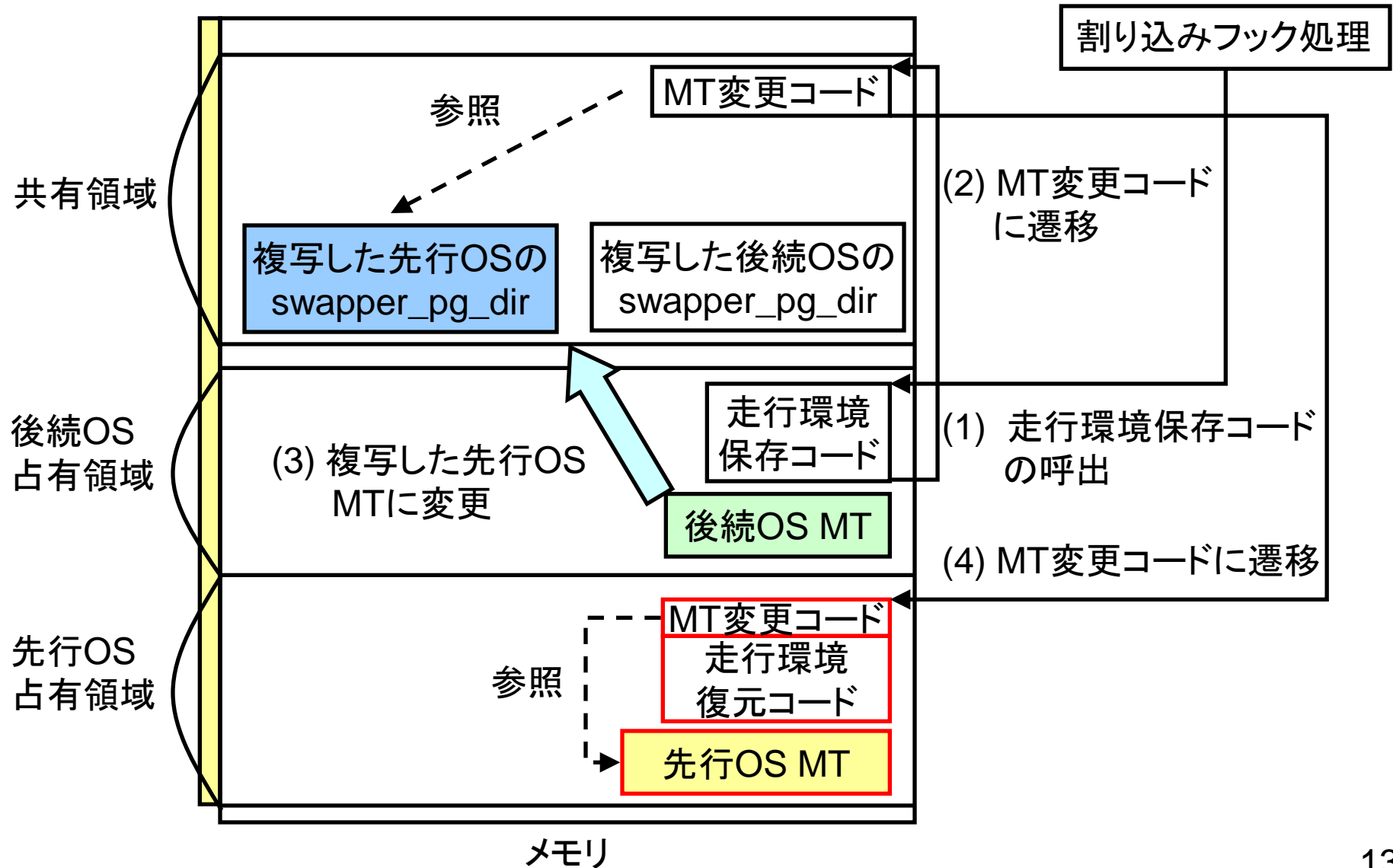
(案1) 共有領域にMTを配置した場合



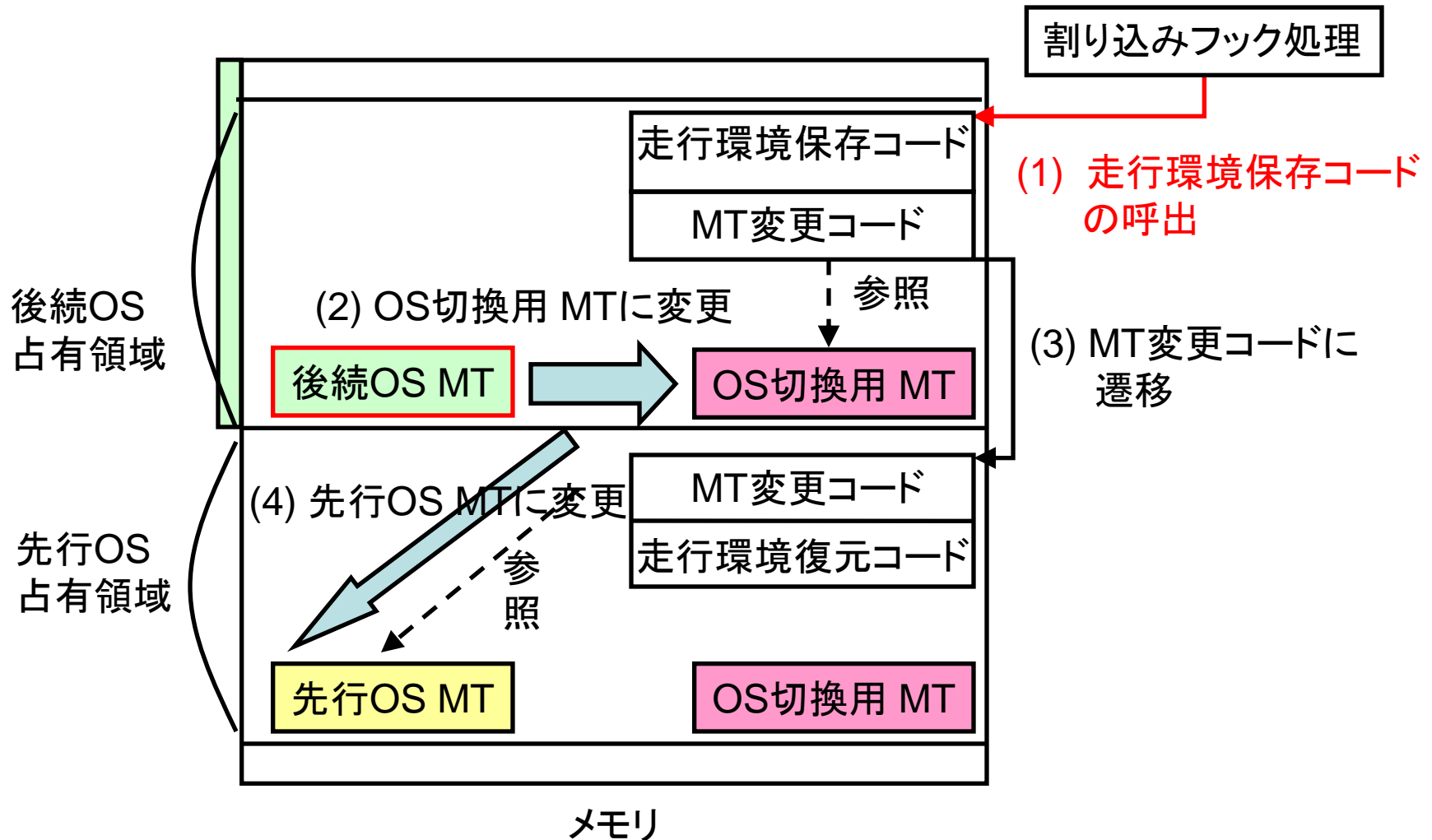
(案1) 共有領域にMTを配置した場合



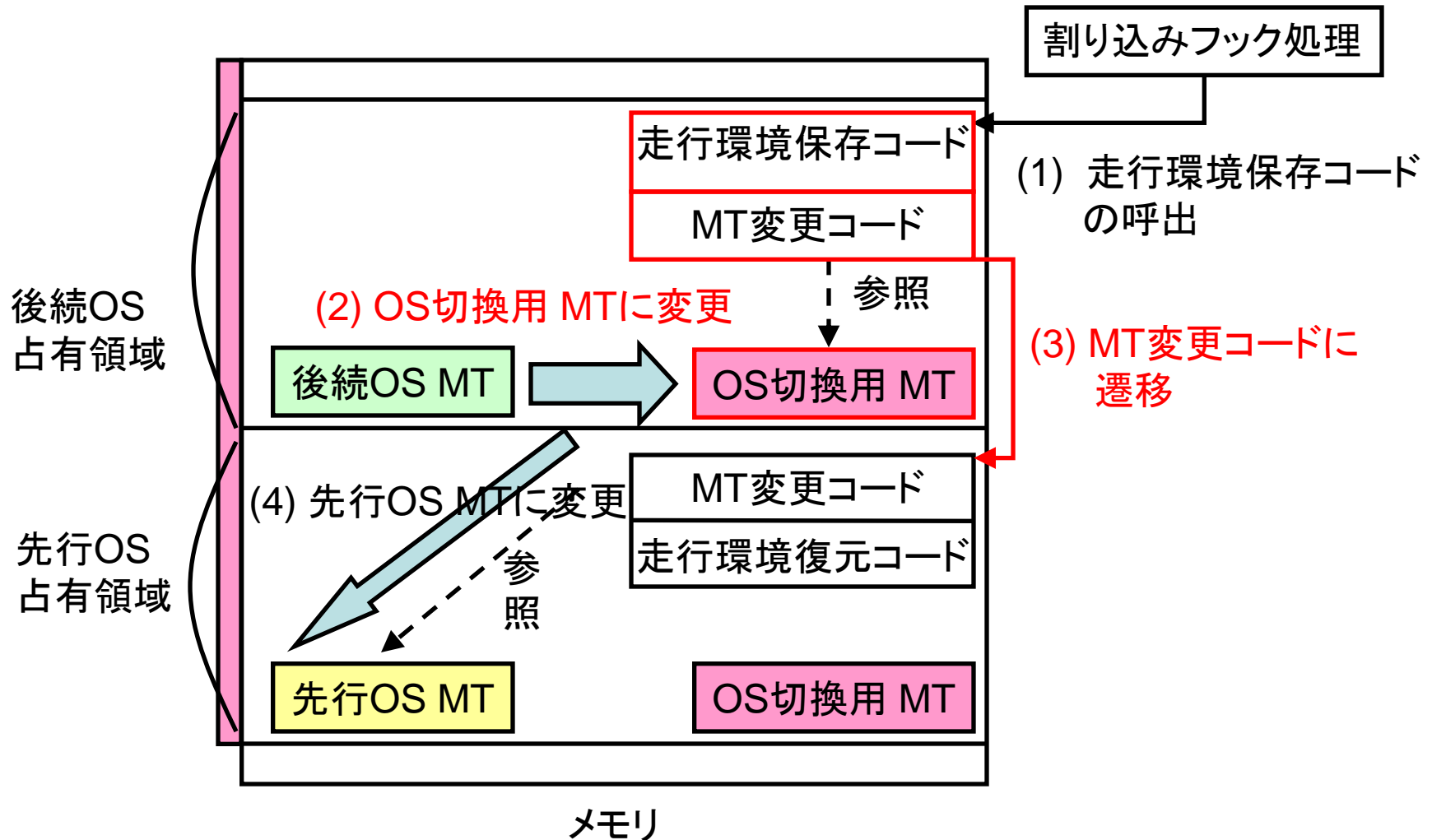
(案1) 共有領域にMTを配置した場合



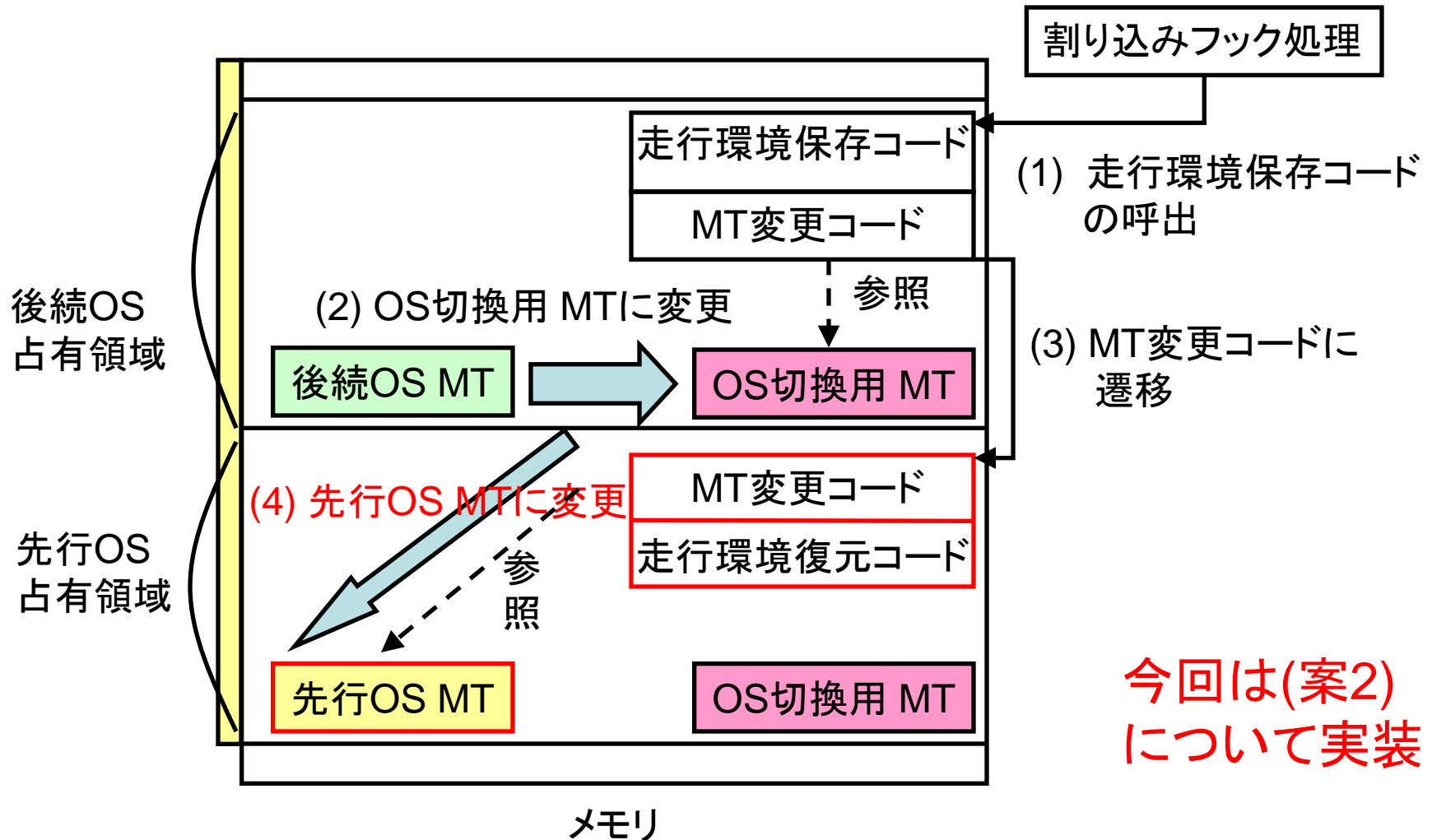
(案2) 各OSの占有領域にMTを配置した場合



(案2) 各OSの占有領域にMTを配置した場合

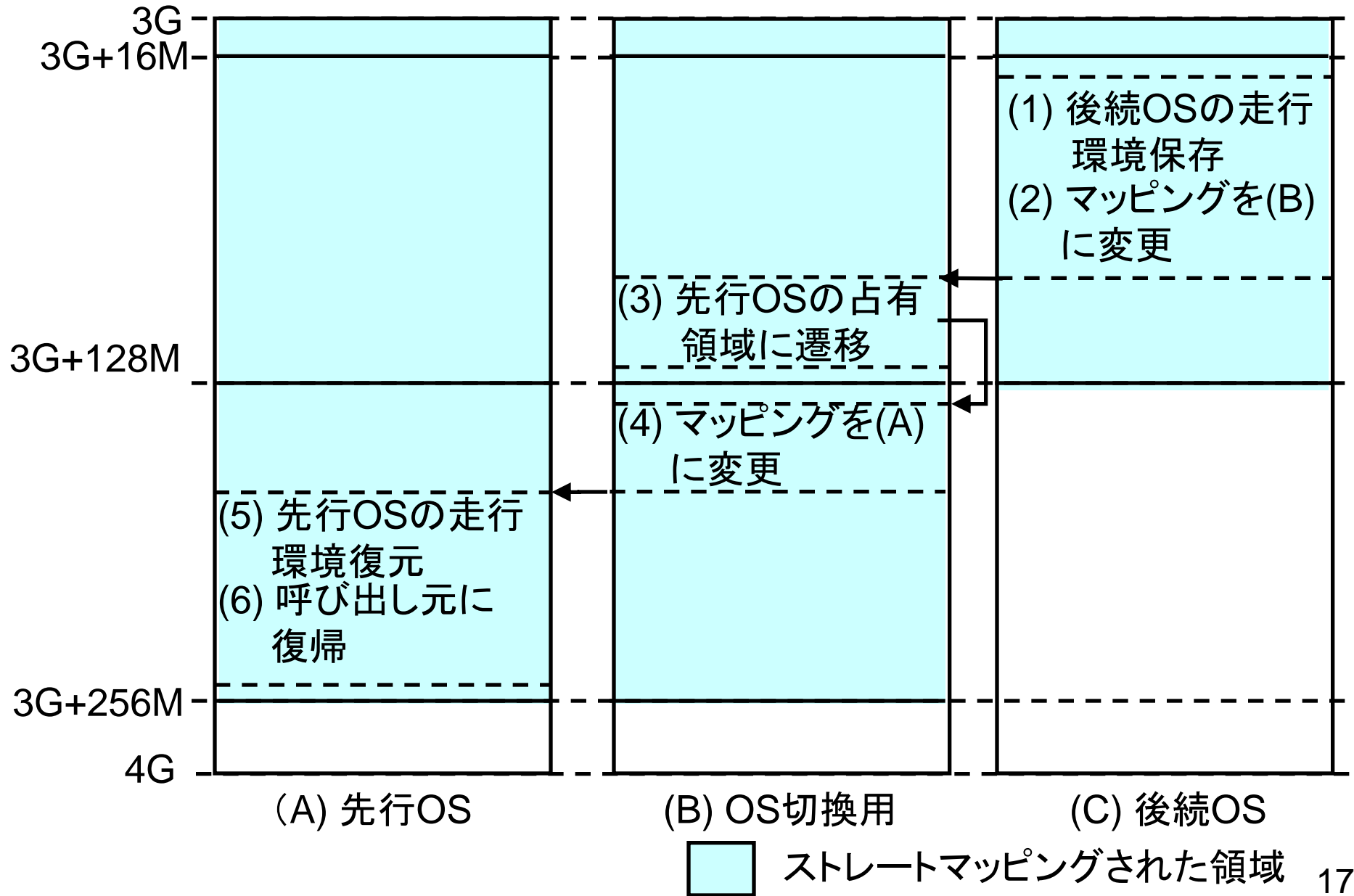


(案2) 各OSの占有領域にMTを配置した場合



今回は(案2)
について実装

OS切換の流れ



おわりに

<複数OS走行環境において計算資源を効率的に利用する方法>

- (1) 併用分割制御法を提案
- (2) シングルコア時分割制御における課題を提示
- (3) シングルコア時分割制御を設計
- (4) シングルコア時分割制御の一部を実装

<残された課題>

- (1) 割り込みのフックの実装
- (2) 3つ以上のOSの走行を実現
- (3) シングルコア時分割制御のマルチコアCPU対応
- (4) コア分割制御と時分割制御の併用利用対応