オペレーティングシステム

第2回

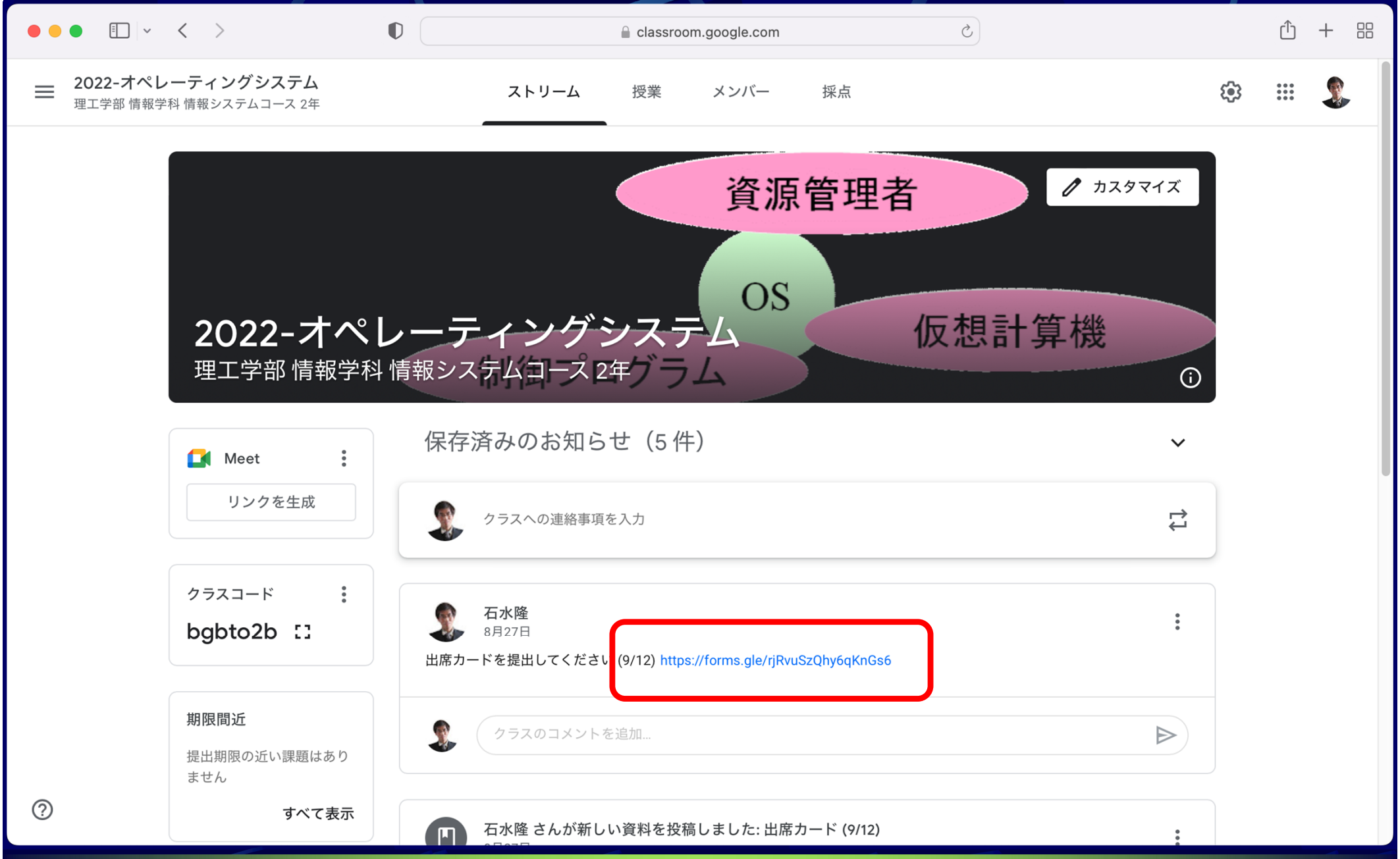
割り込みとOSの構成

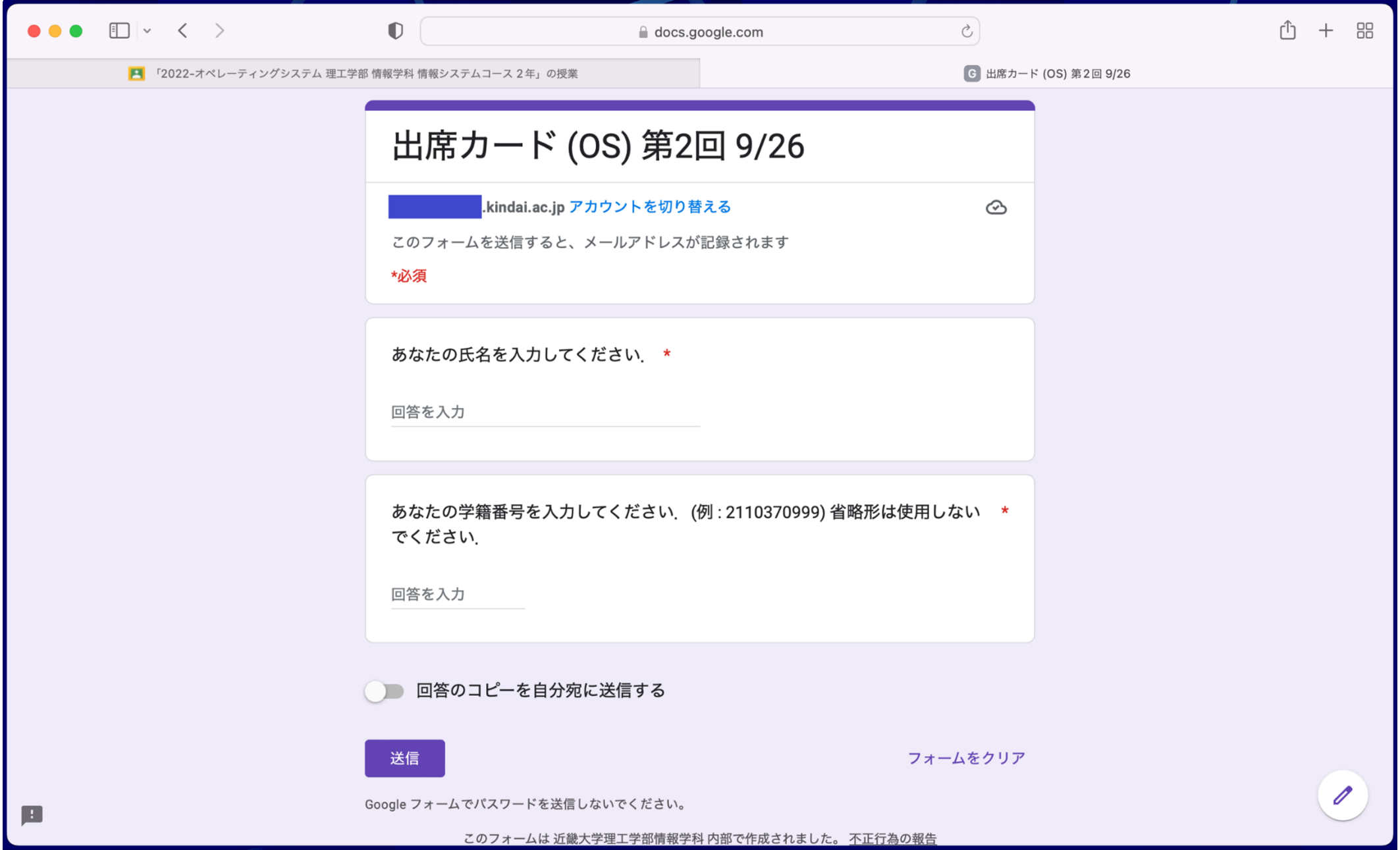
http://www.info.kindai.ac.jp/OS

E館3階E-331 内線5459

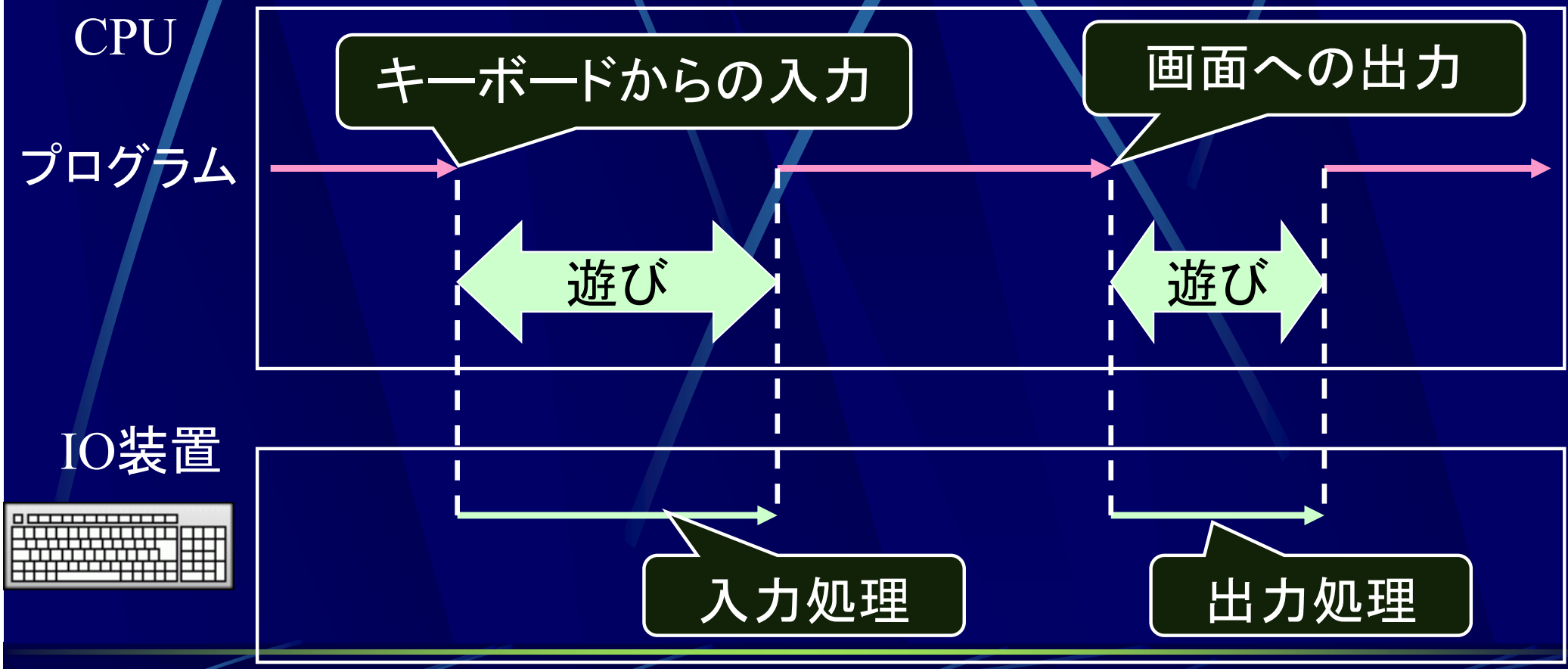
takasi-i@info.kindai.ac.jp

https://classroom.google.com/h


classroom.google.com
2022-オペレーティングシステム
理工学部 情報学科 情報システムコース 2年
ストリーム
授業
メンバー
採点
資源管理者
カスタマイズ
OS
仮想計算機
制御プログラム
2022-オペレーティングシステム
理工学部 情報学科 情報システムコース 2年
Meet
リンクを生成
保存済みのお知らせ（5 件）
クラスへの連絡事項を入力
クラスコード
bgbto2b
石水隆
8月27日
出席カードを提出してください (9/12) https://forms.gle/rjRvuSzQhy6qKnGs6
クラスのコメントを追加...
期限間近
提出期限の近い課題はありません
すべて表示
石水隆 さんが新しい資料を投稿しました: 出席カード (9/12)


docs.google.com
「2022-オペレーティングシステム 理工学部 情報学科 情報システムコース 2年」の授業
出席カード (OS) 第2回 9/26
出席カード (OS) 第2回 9/26
[redacted].kindai.ac.jp アカウントを切り替える
このフォームを送信すると、メールアドレスが記録されます
*必須
あなたの氏名を入力してください. *
回答を入力
あなたの学籍番号を入力してください. (例 : 2110370999) 省略形は使用しないでください. *
回答を入力
回答のコピーを自分宛に送信する
送信
フォームをクリア
Google フォームでパスワードを送信しないでください。
このフォームは 近畿大学理工学部情報学科 内部で作成されました。 不正行為の報告

プログラムの実行中の動作



CPUの遊び時間ができてしまう

単一プログラムの問題点

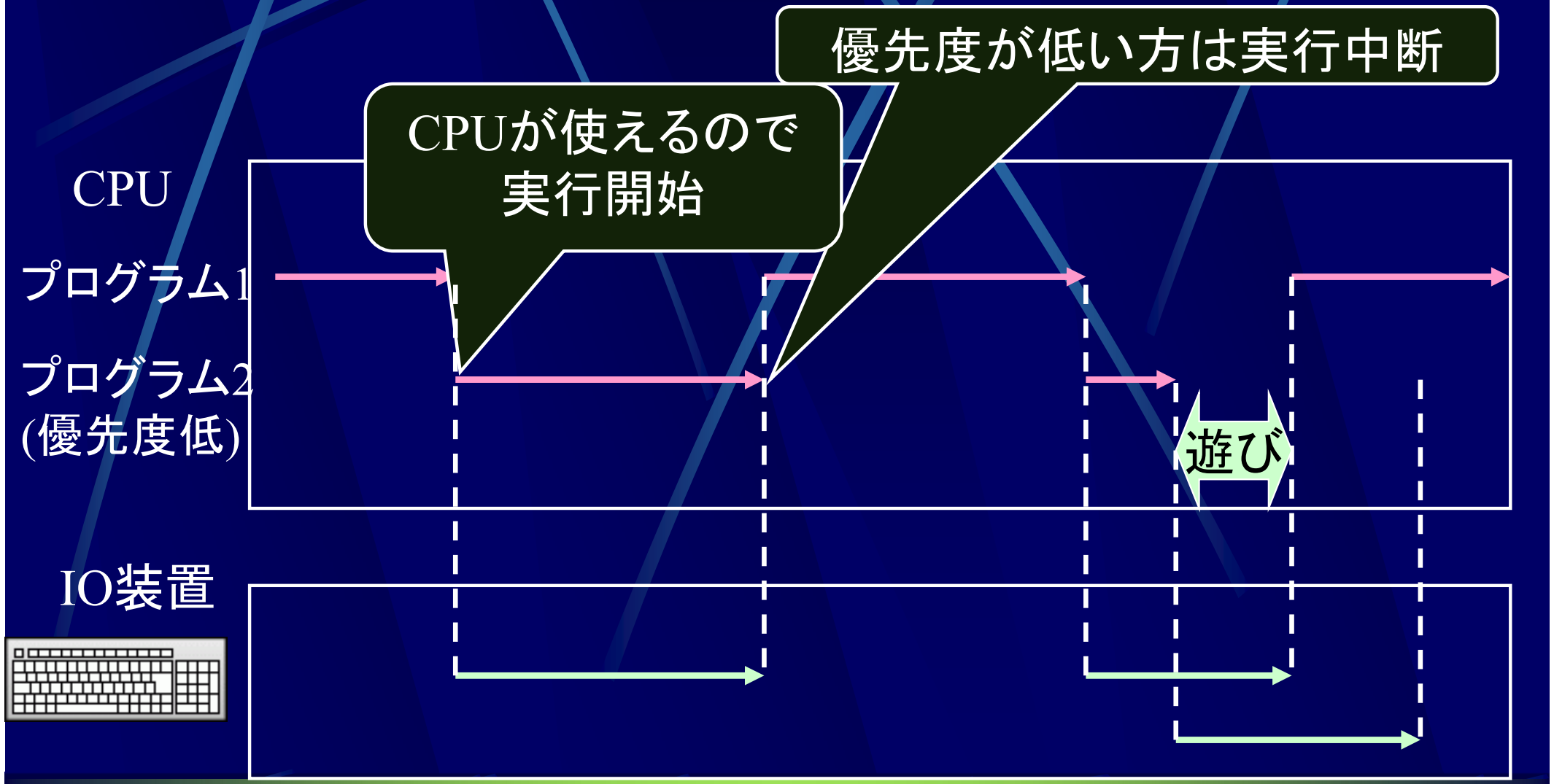
単一プログラムだとCPUの遊び時間が大きくなる

多重プログラムにする

多重プログラム

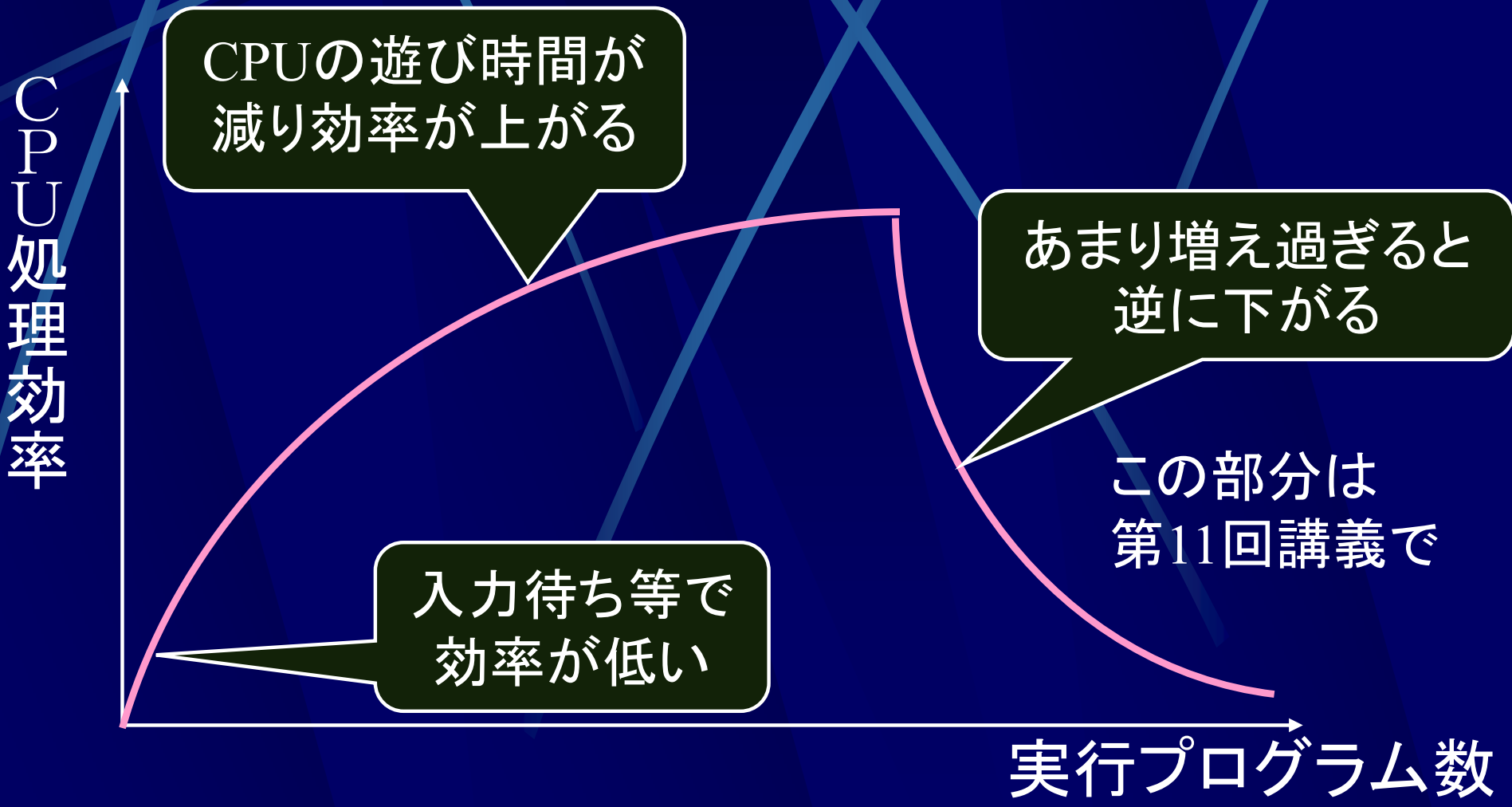
- 複数のプログラムを並行して実行させる

多重プログラムの実行中の動作

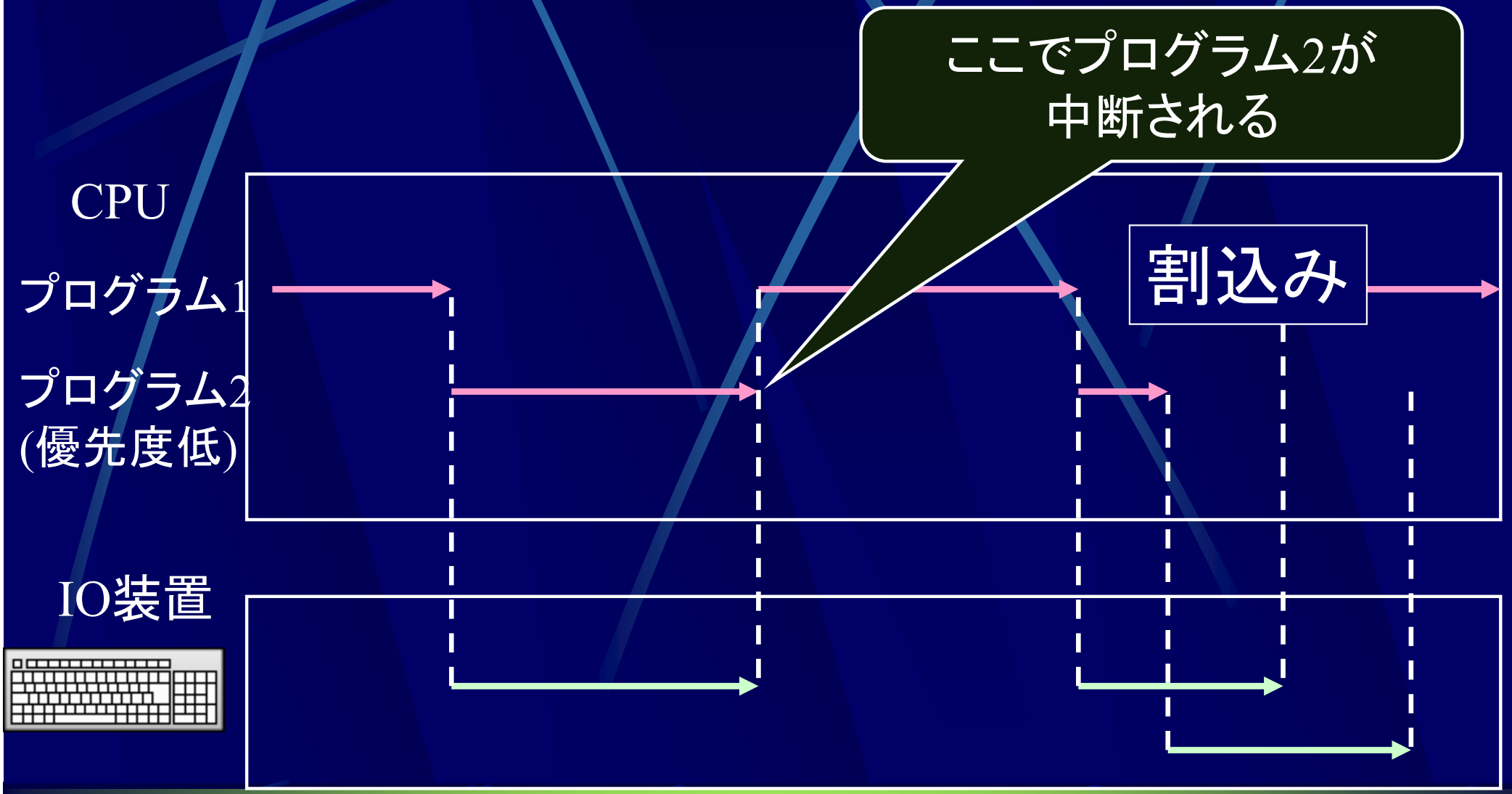


多重プログラムにすればCPUの遊び時間を減らせる

実行プログラム数と処理効率

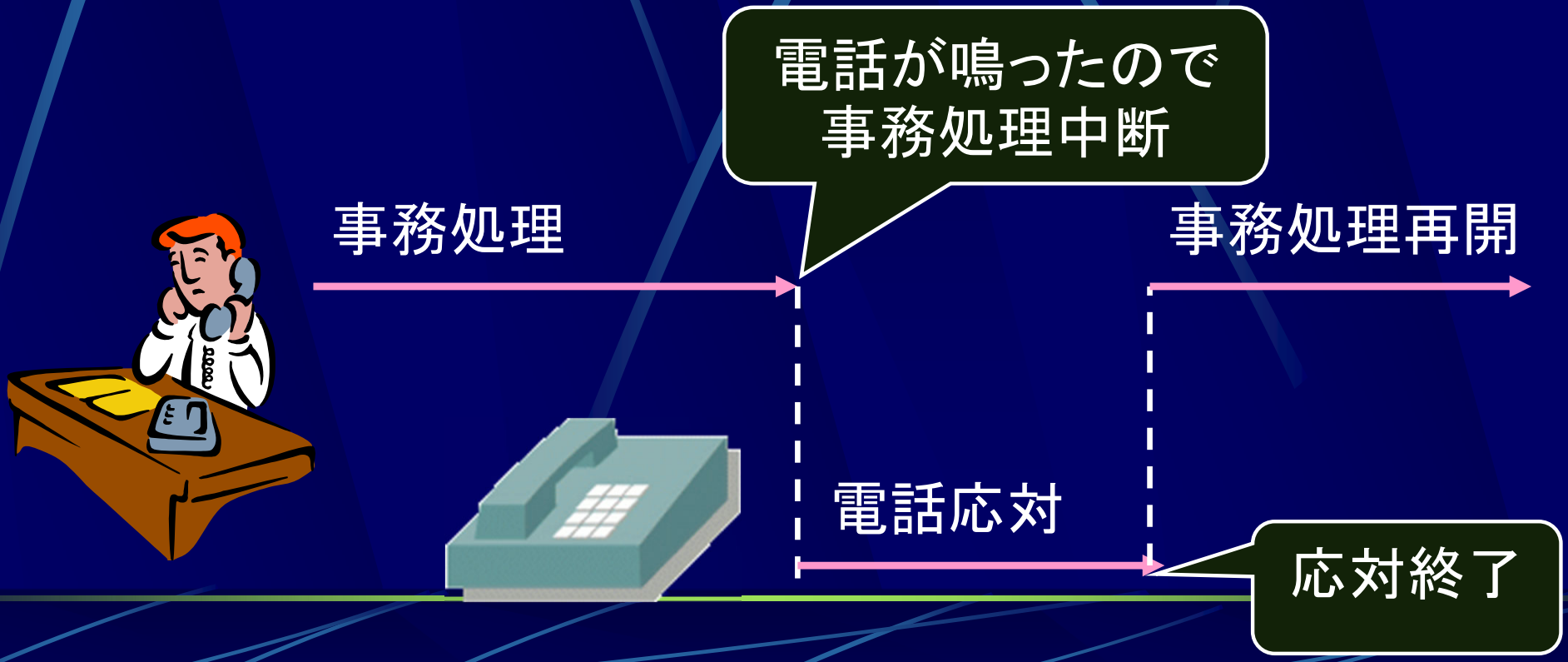


多重プログラムの実行中の動作



割込み(interrupt)

- 割込み(interrupt)
  - 実行中の処理を中断して特別な処理をする

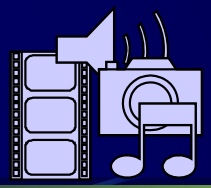


割込みの発生

- 割込みは非同期に発生する
  - 電話はいつ鳴るかは分からない
  - ユーザ入力はいつ完了するかは分からない

事務処理

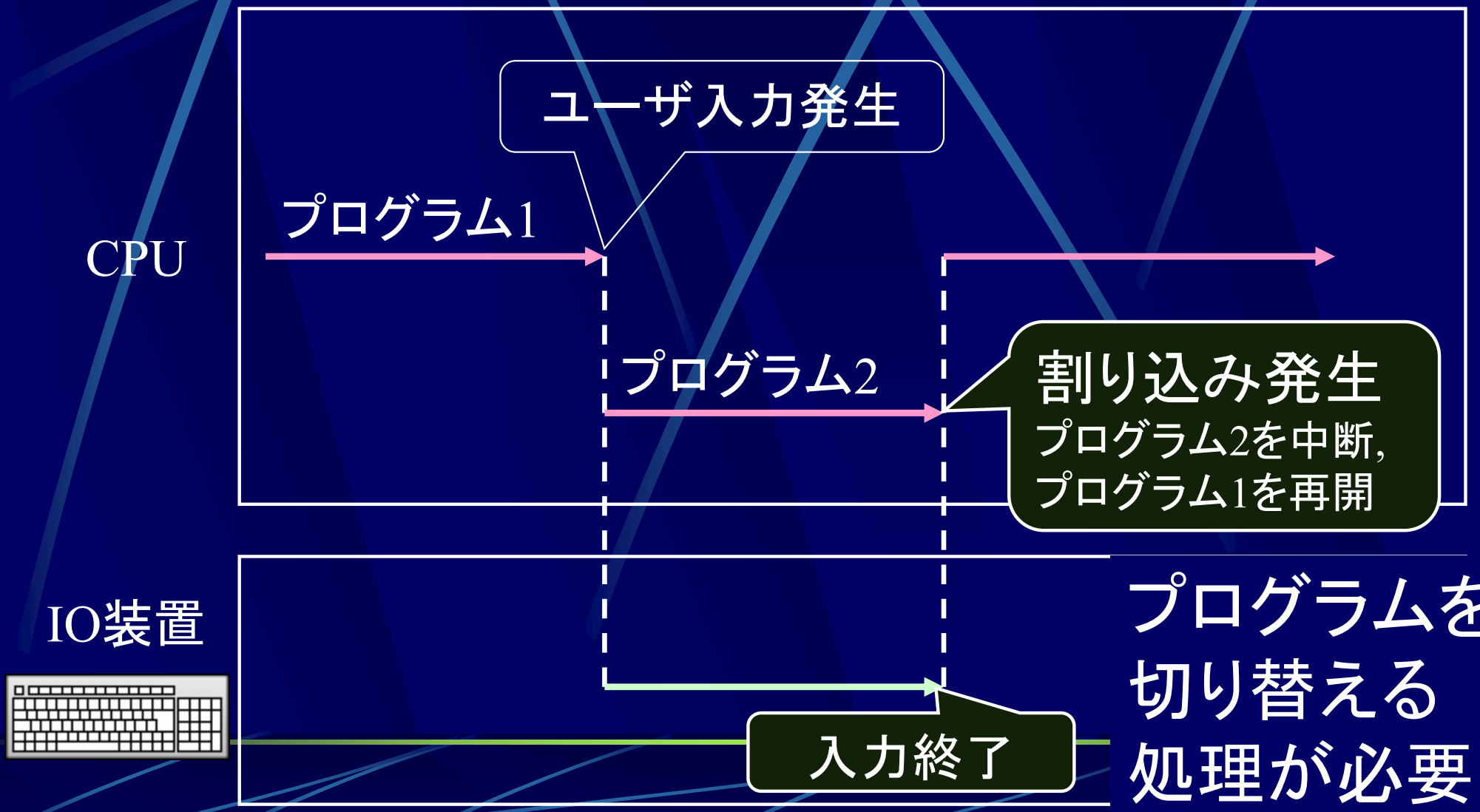
電話は処理中に
突然鳴る



プログラム

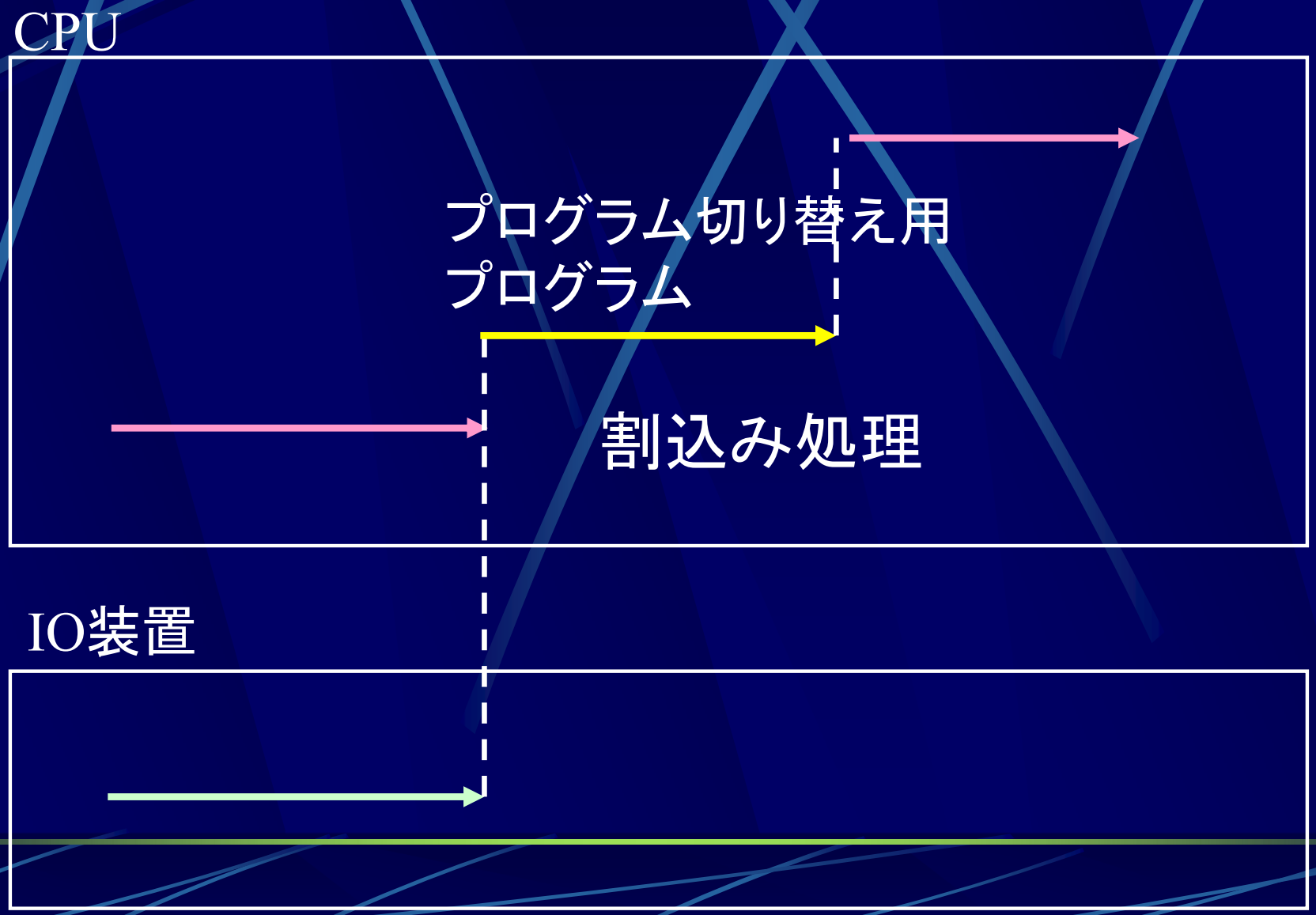
ユーザ入力はプログラム
実行中に突然起きる

割込み

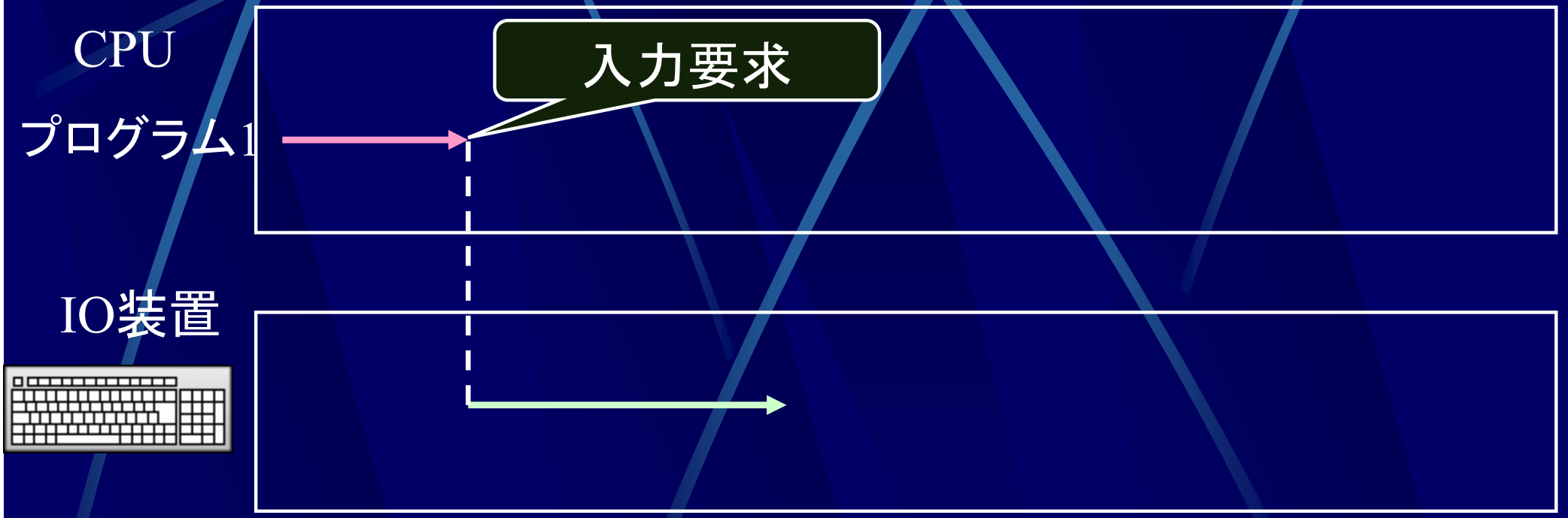


プログラムを
切り替える
処理が必要

割込み



プログラム実行中の入力要求



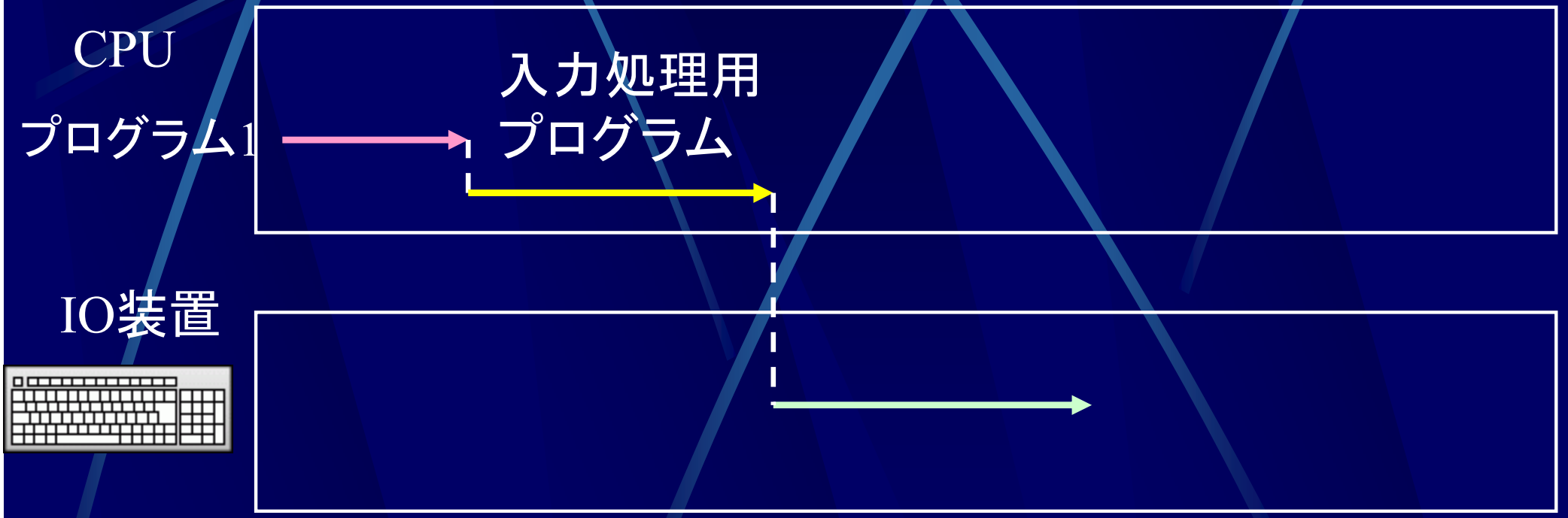
しかし、プログラム1から直接IO装置を起動はできない

プログラム実行中の入力要求

- アプリケーションプログラムが
  直接IO装置を起動することはできない
  - どのプログラムがIO装置を使うかの管理
  - 実行プログラムの切り替え

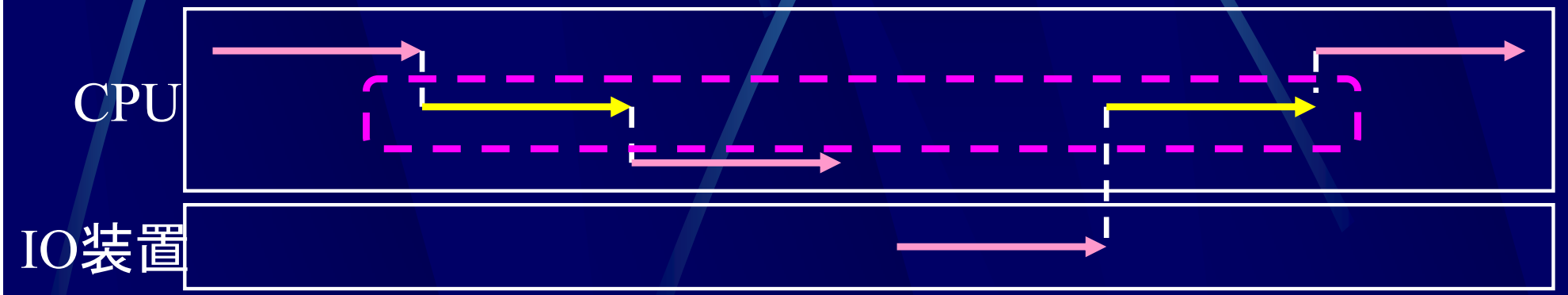
これらを処理するプログラムが必要

プログラム実行中の入力要求



特別な処理を行うプログラム

- 特別な処理を行うプログラム
  - プログラム切り替え用プログラム
  - 入出力処理用プログラム



カーネル(kernel)
OSの中核部分

カーネル(kernel)

- OSの基本的なサービス
  - 資源の割付と保護
  - プログラムの実行
  - 入出力操作
  - ファイル操作

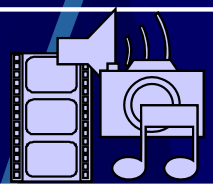
カーネル(kernel)
基本的サービスを行うOSの根幹

制御プログラムとしての役割

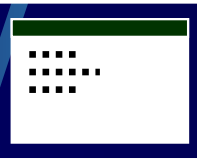
カーネル

ユーザ





アプリケーションプログラム



システムプログラム　エディタ, コンパイラ
リンカ, ローダ 等

カーネル
(kernel)

プロセス管理, 同期と通信制御
ファイルシステム, メモリ管理
スケジューラ, 割り込み制御, 入出力制御
タイマ管理, デバイスドライバ 等

ハードウェア　機械語, 物理デバイス
マイクロプログラム 等

狭義のOS

広義のOS

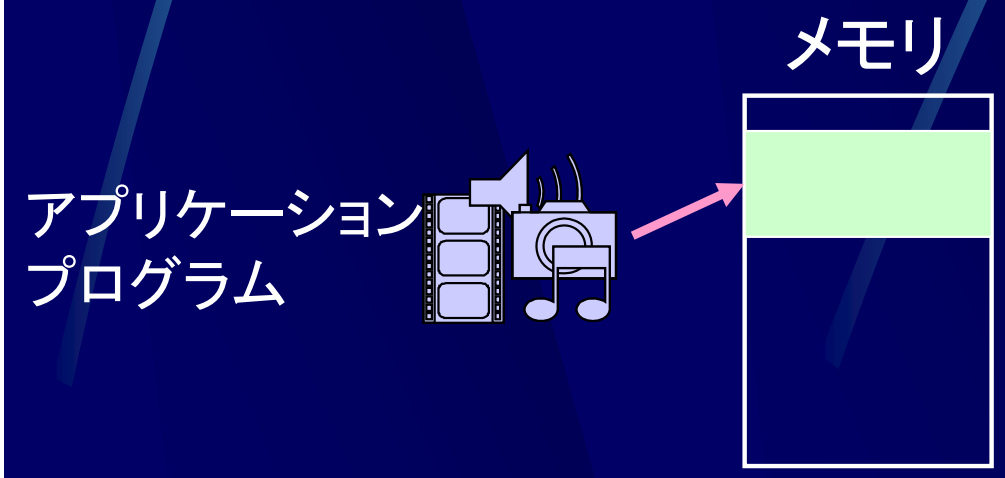
カーネルの特徴

- カーネルの特徴
  - 割込みにより起動
  - カーネルモード(特権モード)で動作
  - アプリケーションプログラムから記憶保護
  - プログラム実行の管理
  - 計算機資源の管理
  - 特別な特権命令でアプリケーションプログラムに戻る

カーネルモード,特権モード
(kernel mode, privileged mode)

- 計算機全体に影響を与える命令を実行できるモード
  - 資源の管理, 制御

例 : メモリ管理



アプリケーションプログラムが
直接メモリアクセスするのは禁止

他のプログラムの領域を
書き換えられる危険

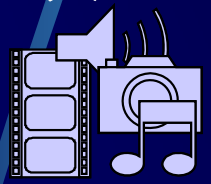
アプリケーションプログラムに
メモリ管理という余計な負荷

メモリ管理はOSが行う

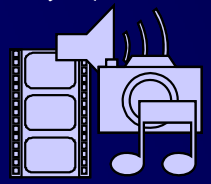
資源の管理，制御

例：メモリ保護

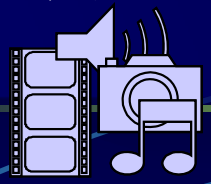
アプリケーション1

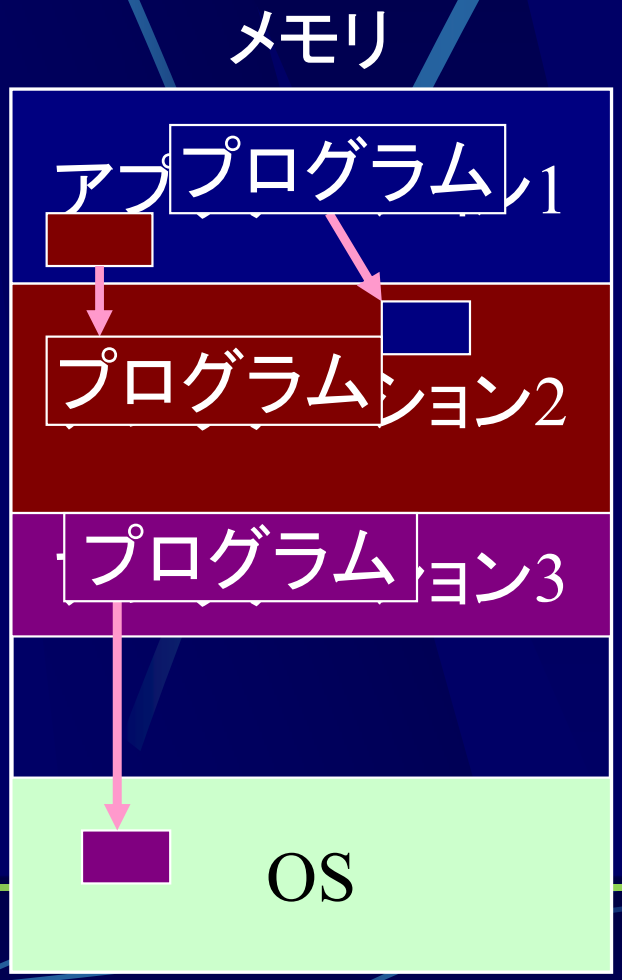


アプリケーション2



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

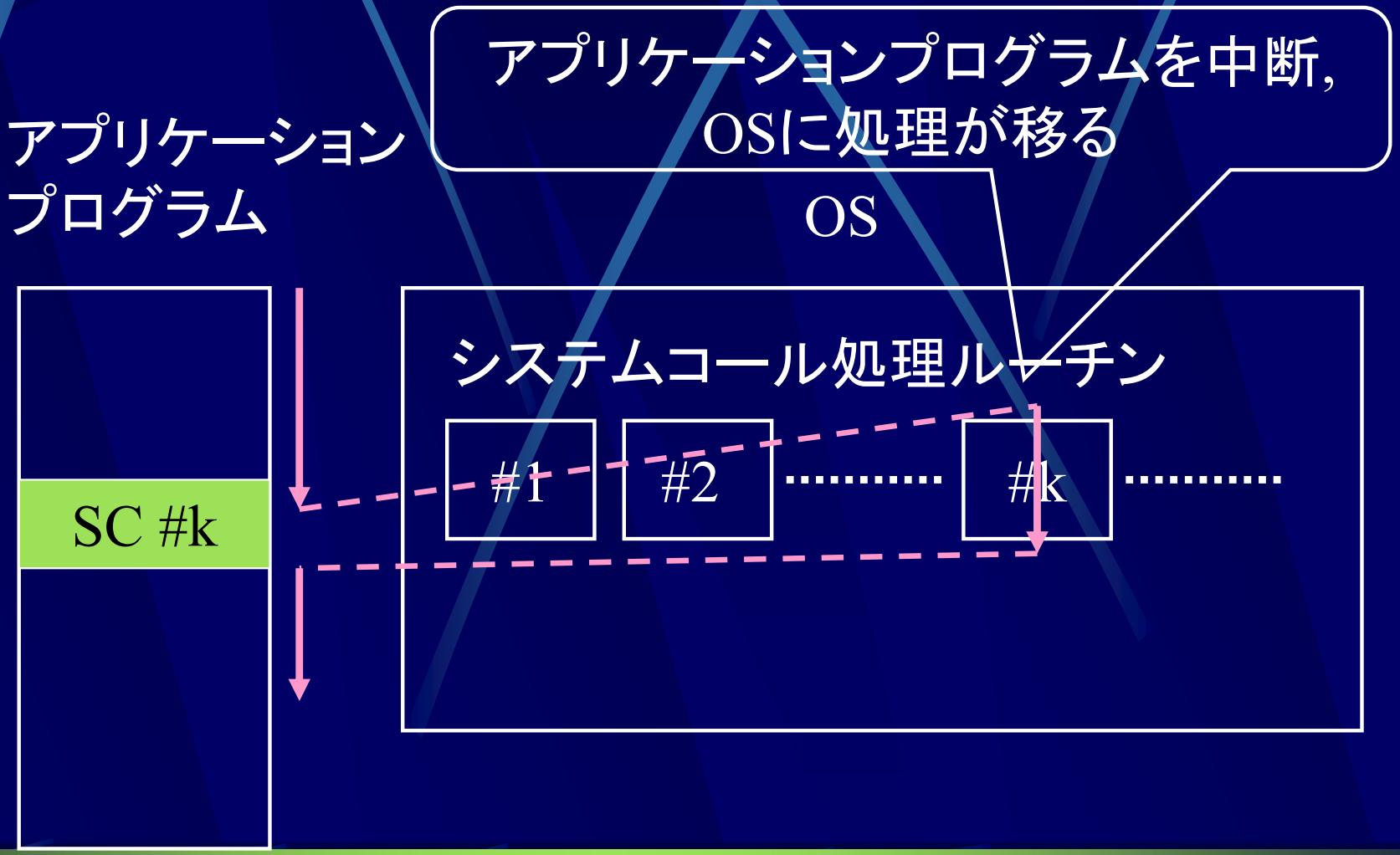
3がOSの領域に
不当な書き込み

アプリケーションを
停止する

システムコール(system call)

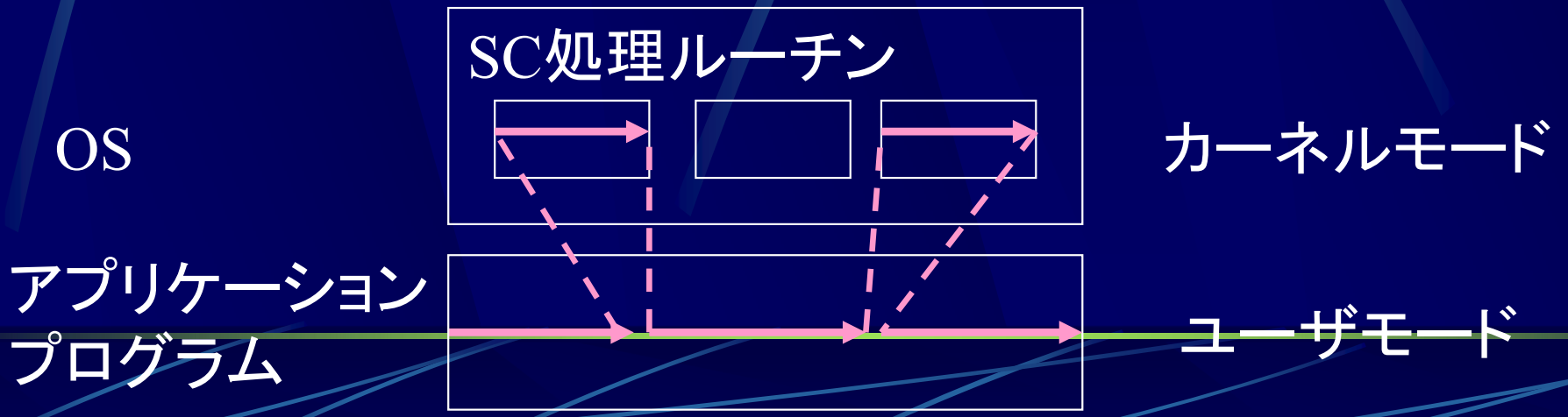
- アプリケーションプログラムから
  OSへのサービス要求
  - 入出力をしたい
  - 他のプログラムを起動・停止したい
  - ファイルに読み・書きしたい
- 手続き呼び出しの形式を取る

システムコール


アプリケーションプログラムを中断,
OSに処理が移る
アプリケーション
プログラム
OS
システムコール処理ルーチン
#1
#2
#k
SC #k

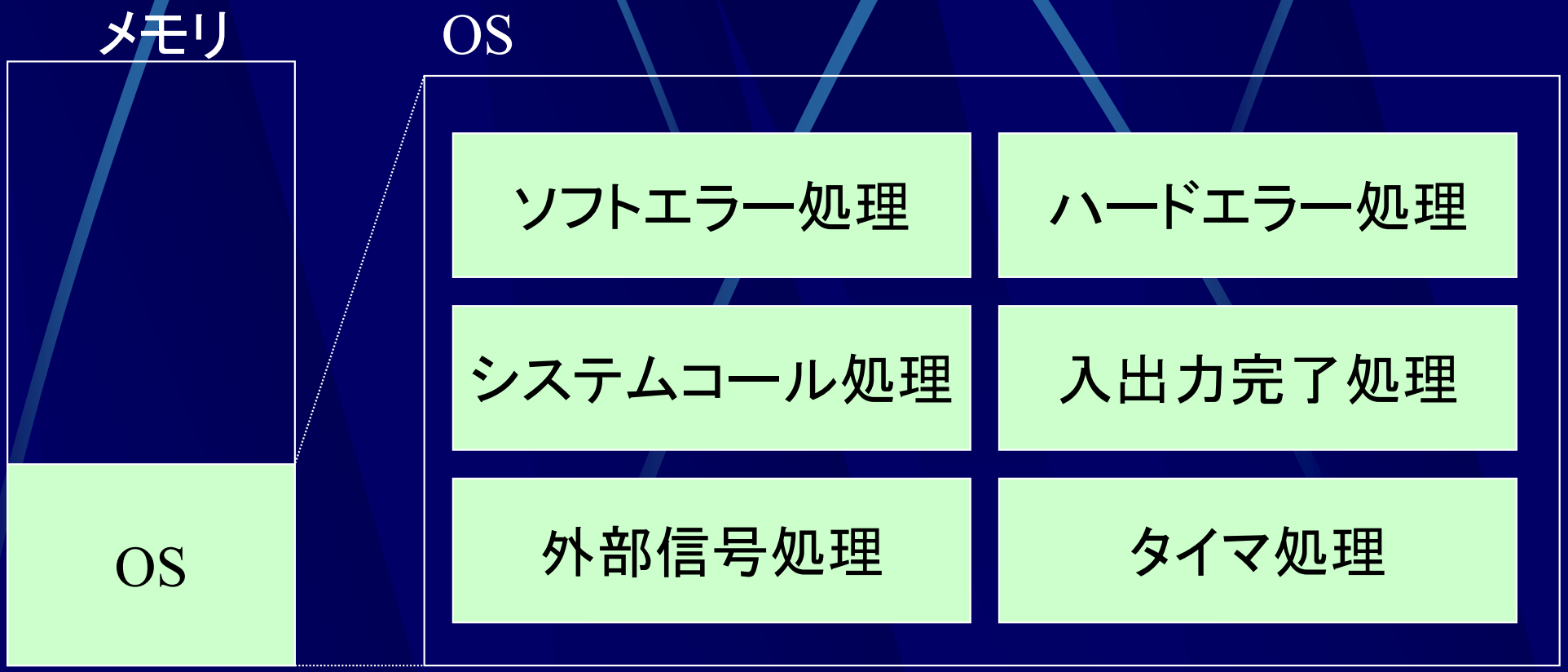
カーネルモード，ユーザモード
(kernel mode, user mode)

- カーネルモード
  - システムコール等の割り込みにより発生，各種割込み処理ルーチン実行中
- ユーザモード
  - アプリケーションプログラムを実行中

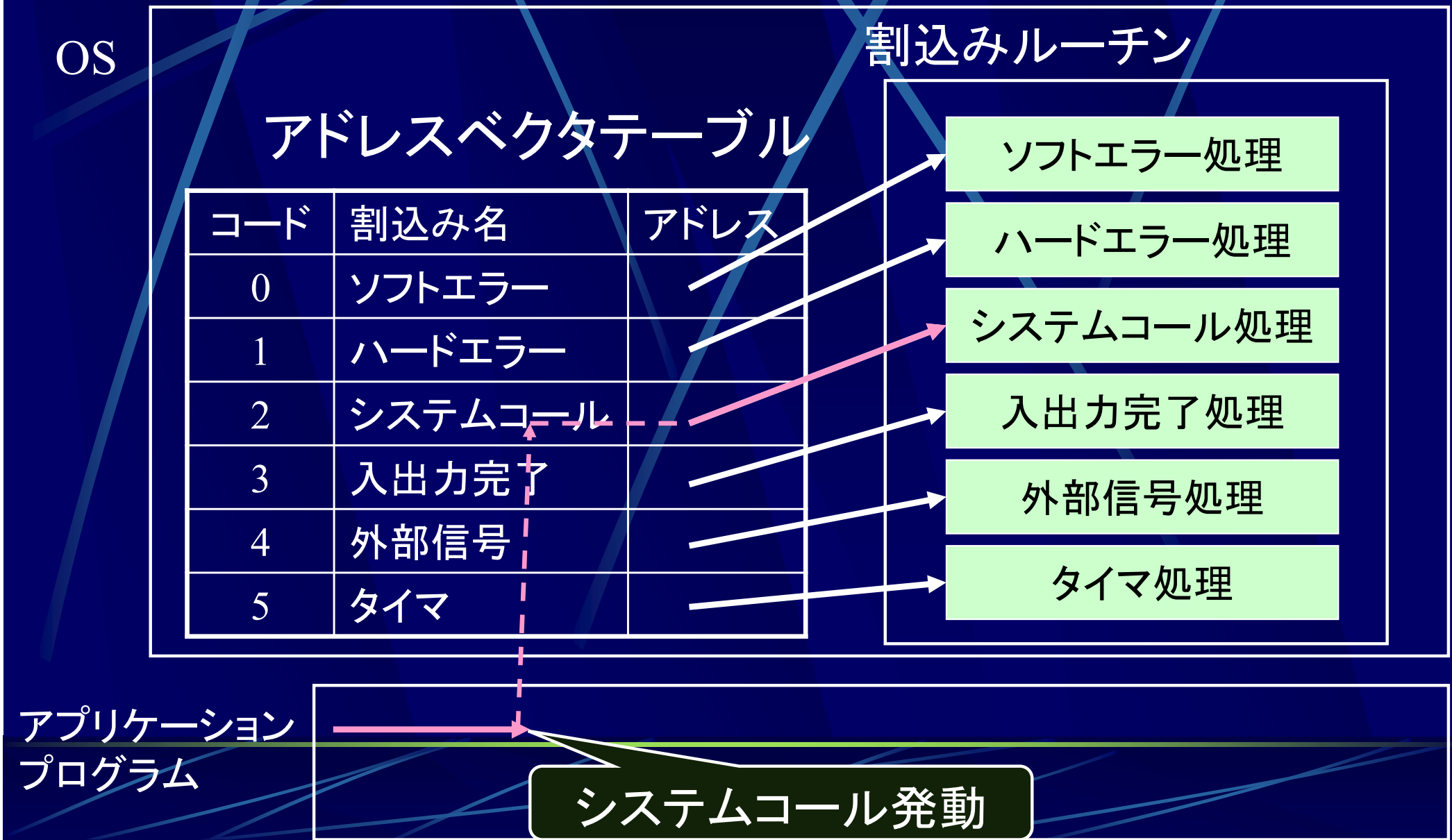


割込み

- メモリのOS領域内に割込みの処理ルーチン



割込み

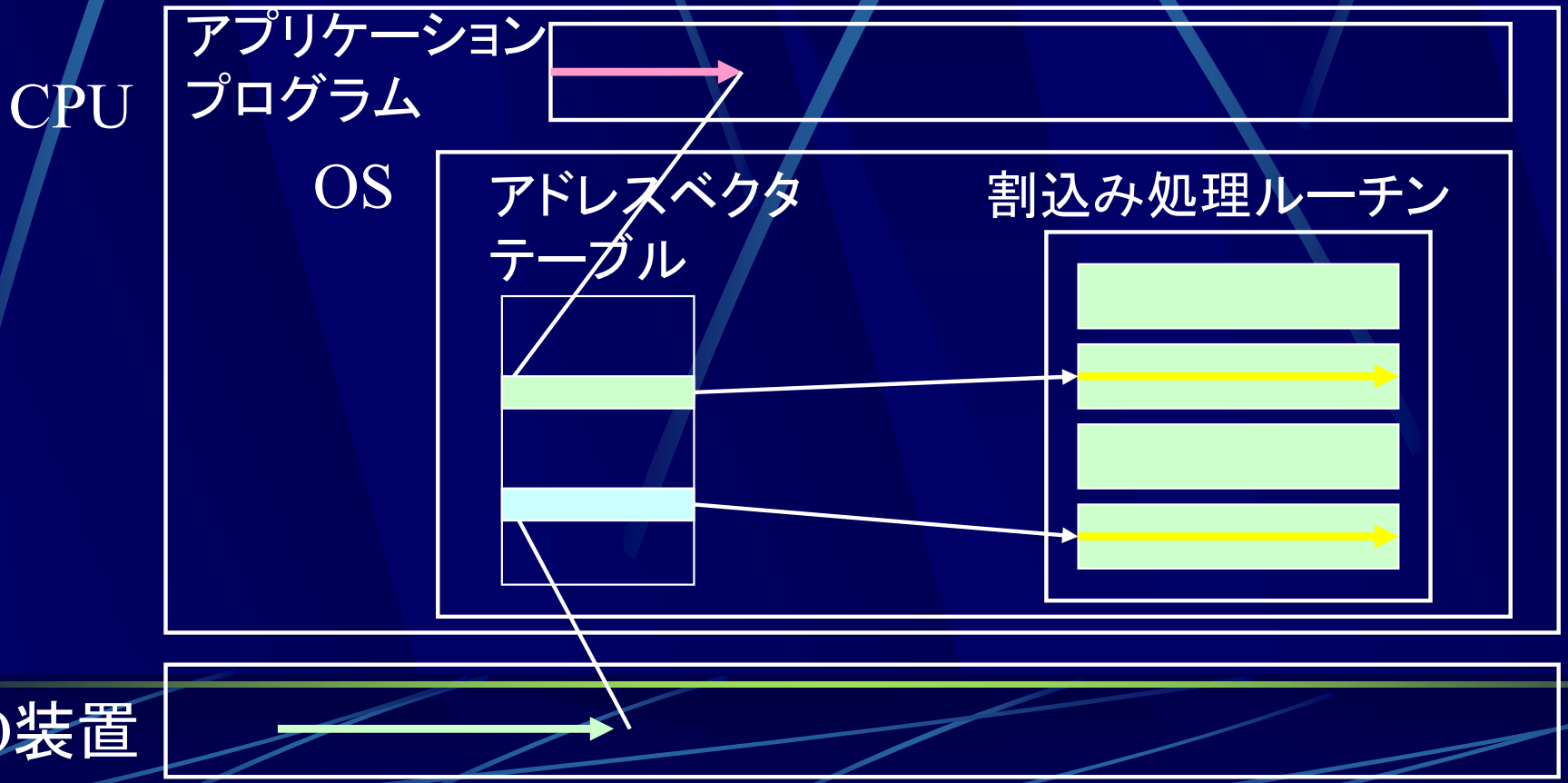

OS
アドレスベクタテーブル
コード
割込み名
アドレス
0
ソフトエラー
1
ハードエラー
2
システムコール
3
入出力完了
4
外部信号
5
タイマ
割込みルーチン
ソフトエラー処理
ハードエラー処理
システムコール処理
入出力完了処理
外部信号処理
タイマ処理
アプリケーション
プログラム
システムコール発動

カーネルの構成要素

1. 割り込み制御
2. 入出力制御
3. タイマ管理
4. 記憶管理
5. CPUスケジューラ
6. プロセス管理
7. 同期と通信制御
8. ファイルシステム

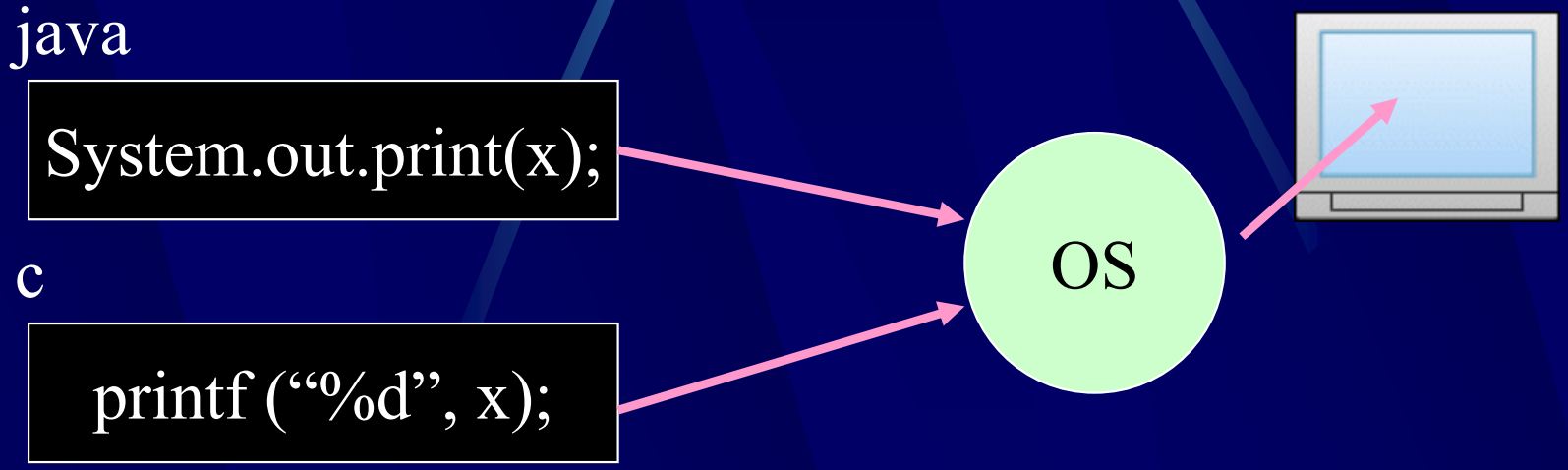
カーネルの構成要素
割込み制御

1. 割込み制御
   - 割り込み要因の解析と処理ルーチンへの分岐



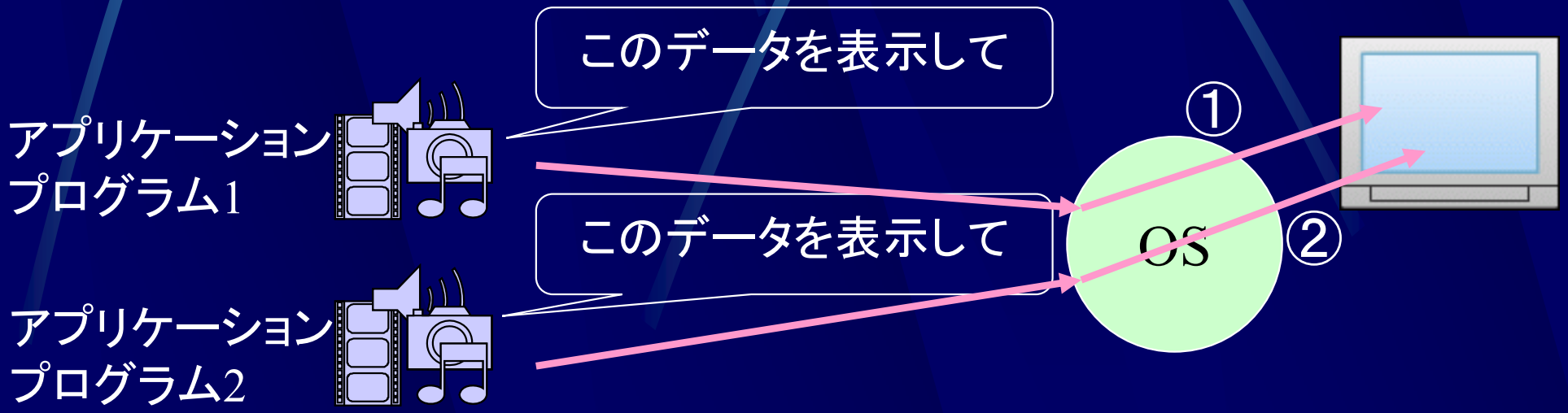
カーネルの構成要素
入出力制御

2. 入出力制御
   - 入力動作のスケジューリング
   - 入出力装置の仮想化



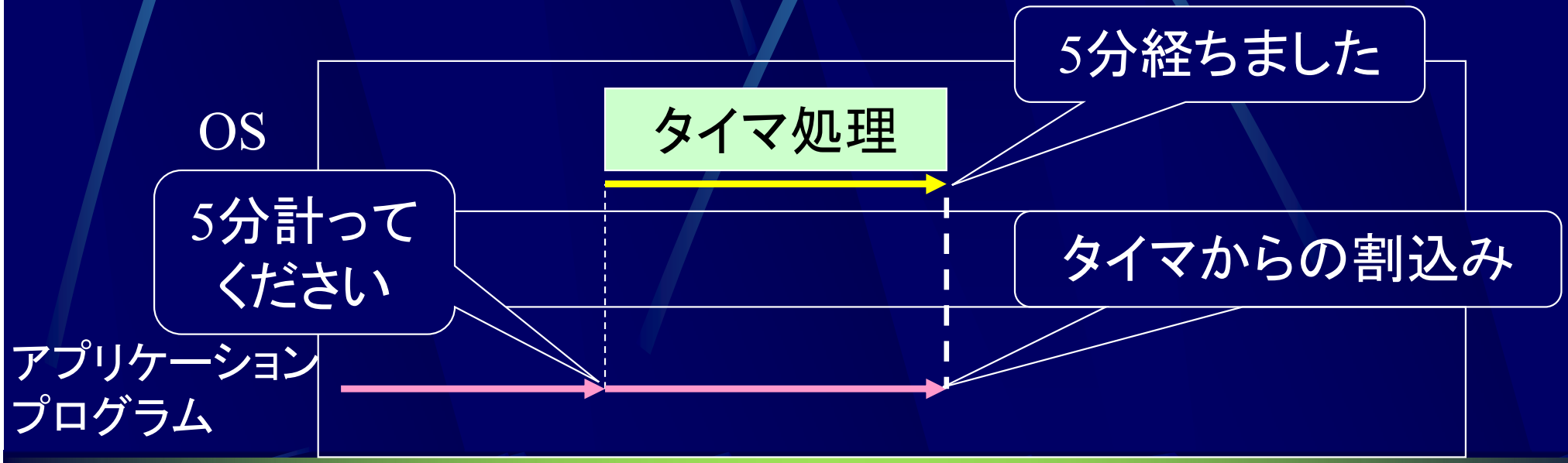
カーネルの構成要素
入出力制御

2. 入出力制御
   - 入力動作のスケジューリング
   - 入出力装置の仮想化



カーネルの構成要素
タイマ制御

3. タイマ制御
   - 時刻の管理
   - 時間経過の監視, 通知など

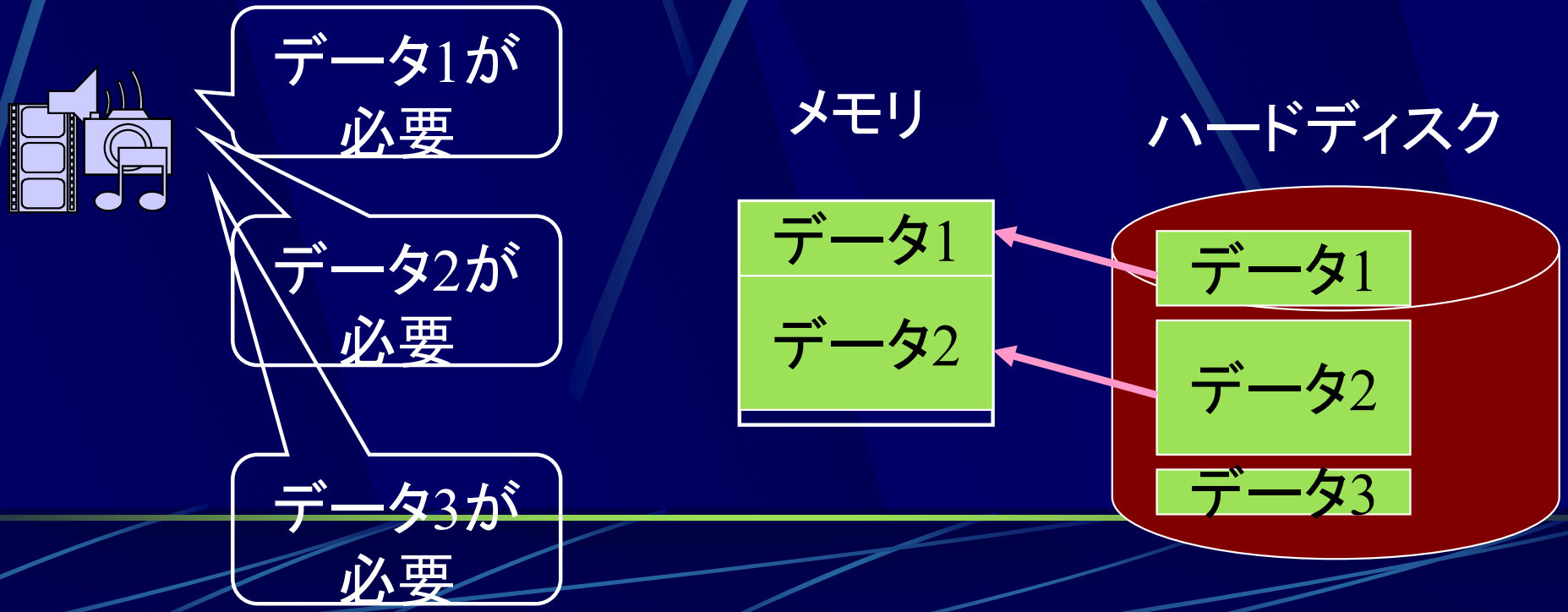


カーネルの構成要素
記憶管理

4. 記憶管理
   - 主記憶管理:メモリ割り付け
   - 仮想記憶管理:ページング，動的置き換え

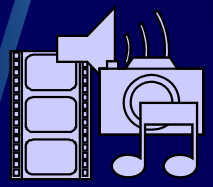
カーネルの構成要素
記憶管理

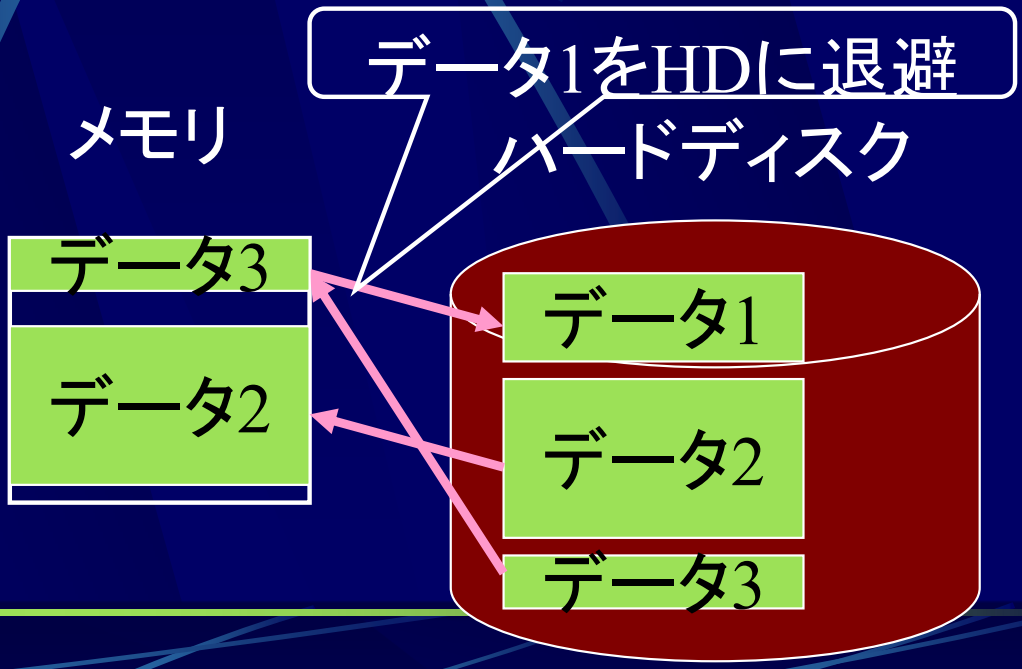
4. 記憶管理
   - 主記憶管理:メモリ割り付け
   - 仮想記憶管理:ページング，動的置き換え



カーネルの構成要素
記憶管理

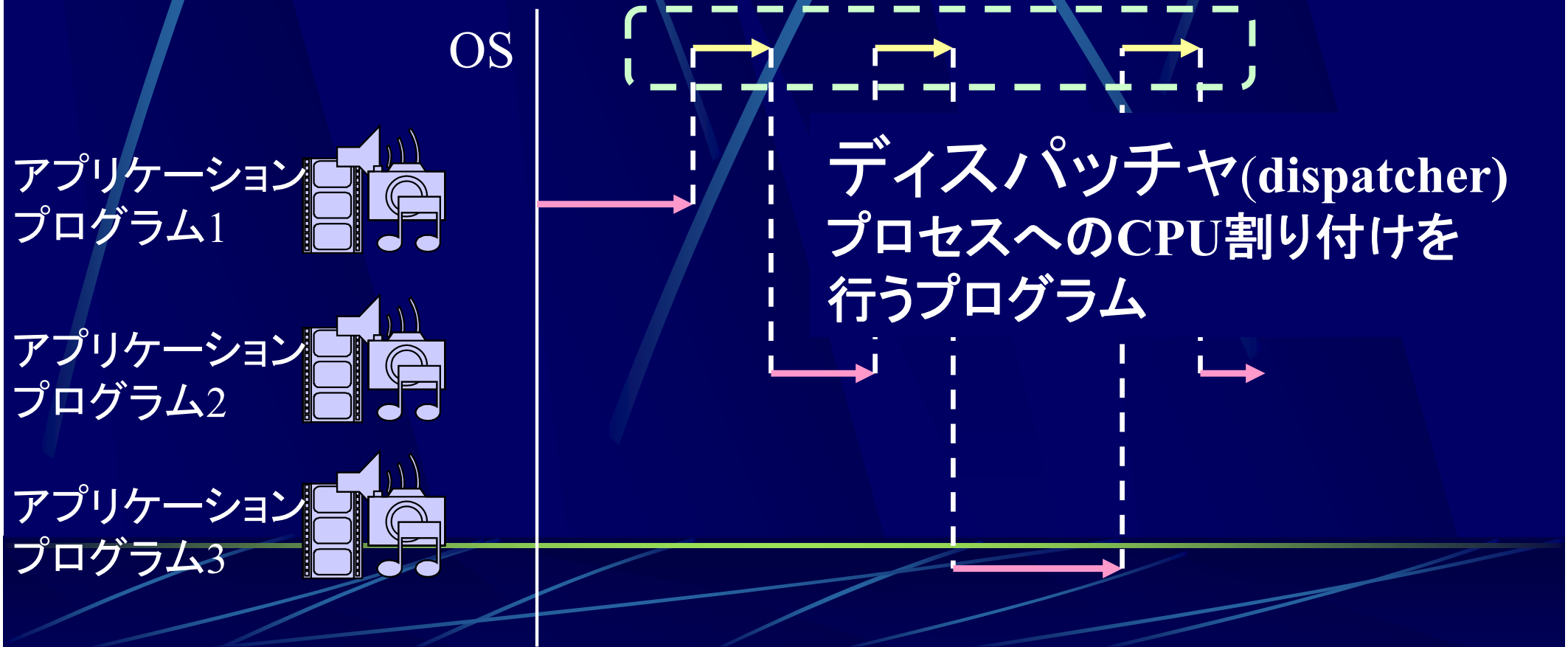
4. 記憶管理
   - 主記憶管理:メモリ割り付け
   - 仮想記憶管理:ページング，動的置き換え





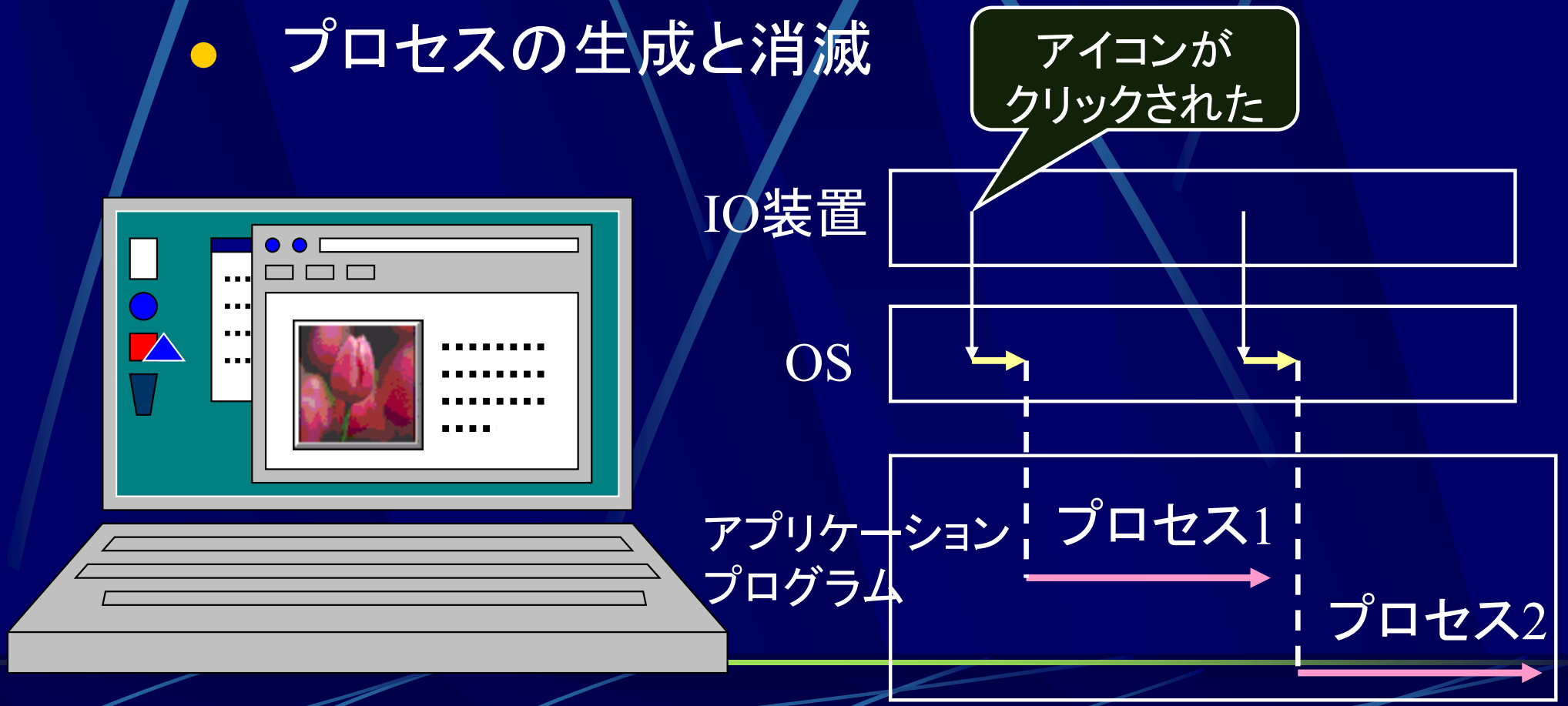
カーネルの構成要素
CPUスケジューラ

5. CPUスケジューラ
   - CPUのプロセスへの割り付けを管理



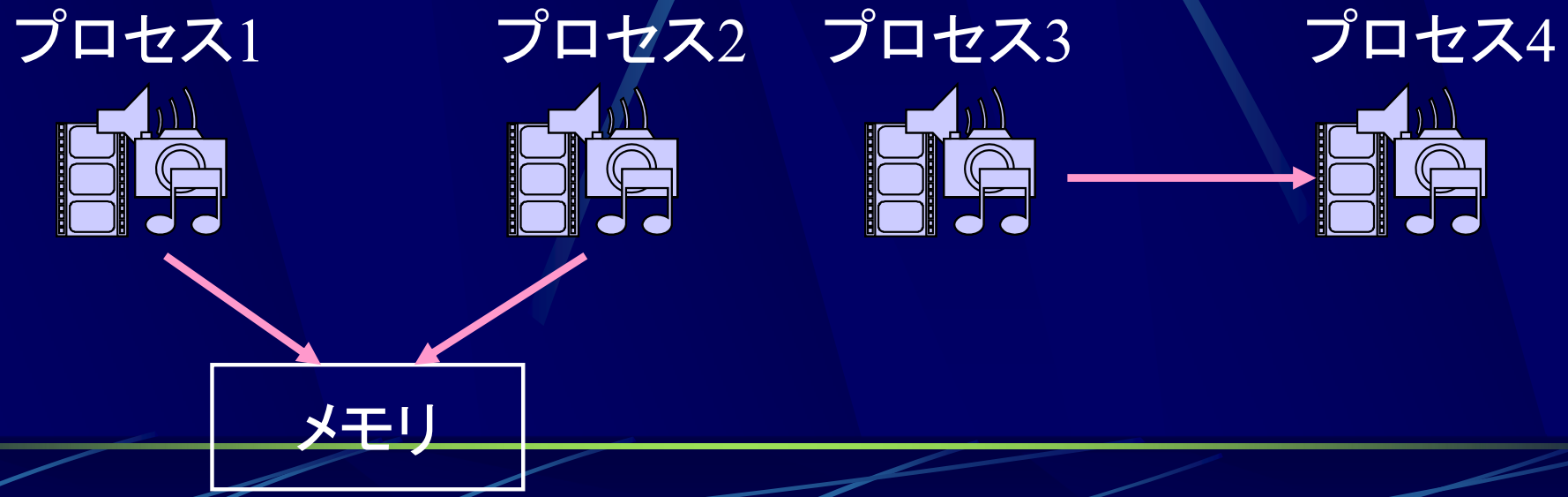
カーネルの構成要素
プロセス管理

6. プロセス管理
   - プロセスの生成と消滅



カーネルの構成要素
同期と通信制御

7. 同期と通信制御
   - 並行プロセス間の協調処理
     - プロセス間同期：排他制御, read-writeの同期
     - プロセス間通信：メッセージ転送

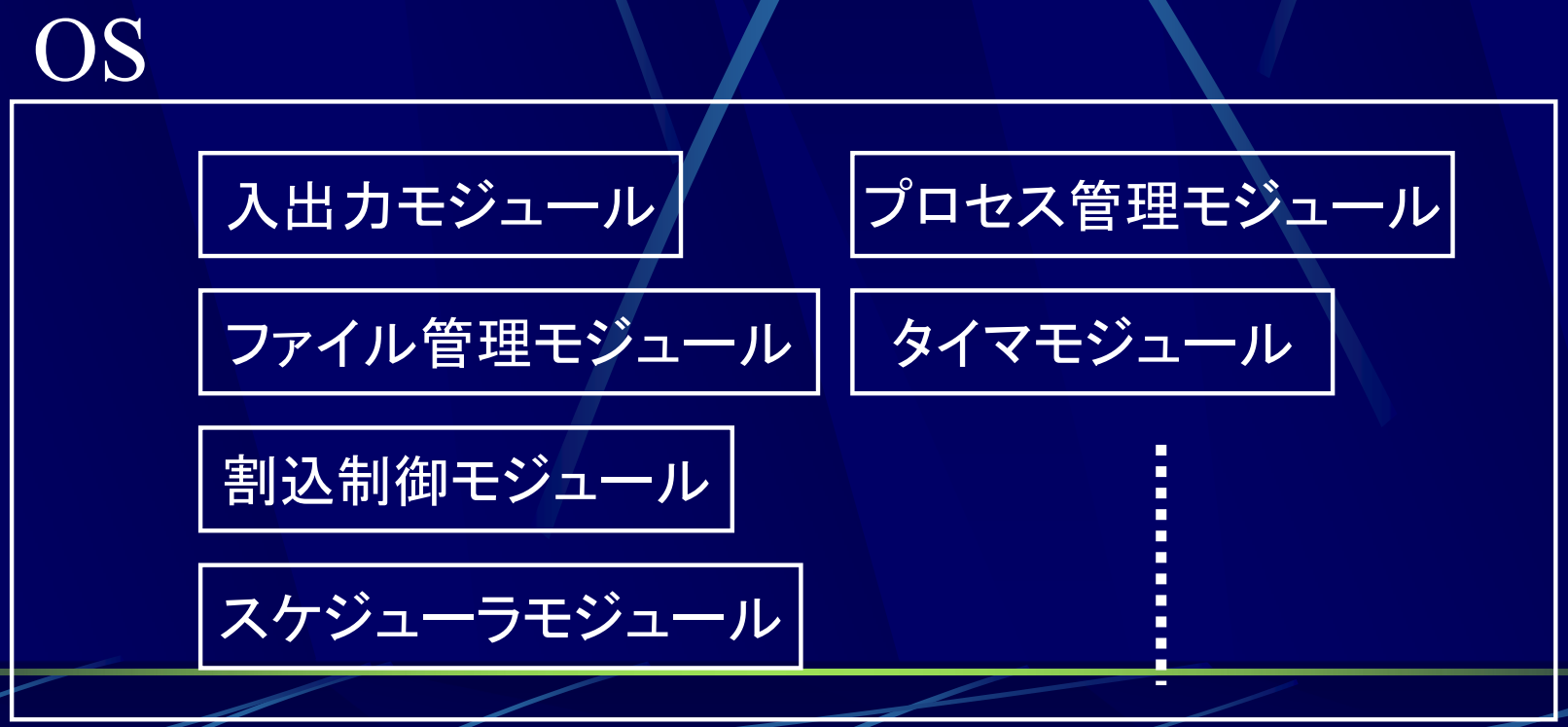


カーネルの構成要素
ファイルシステム

8. ファイルシステム
   - ファイル制御の規定と提供
     - ファイル構造：バイト列/レコード列, キー
     - アクセス法：順アクセス, ランダムアクセス
   - ディレクトリ管理
   - ファイル保護
     - 領域割り付け
     - 一貫性制御

OSの実現法

- OSは多数のモジュールから構成される

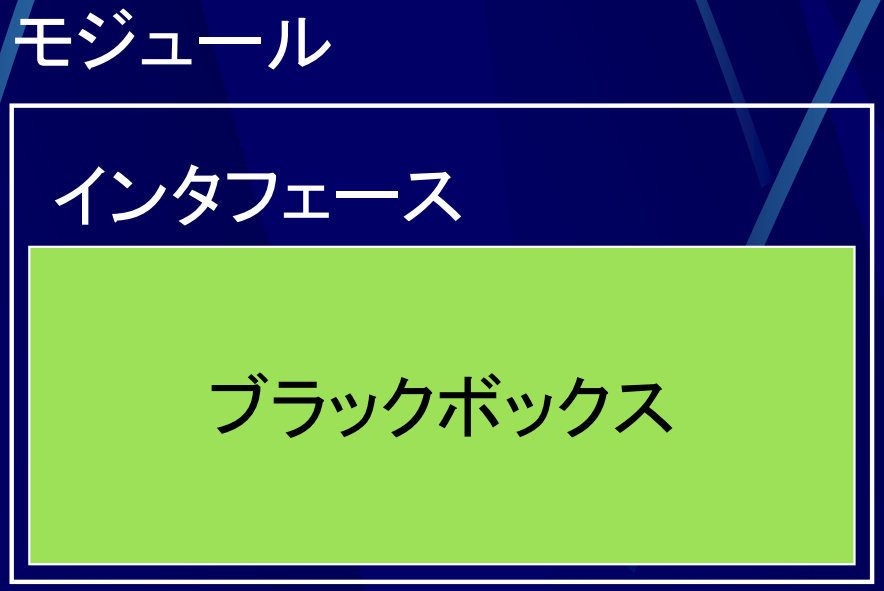


OSの実現法
モジュール化

- モジュール化の基準
  - 情報隠蔽(information hiding)
  - 方針と機構の分離
  - 階層化(Layering)

OSの実現法
情報隠蔽

- モジュール内部の情報を隠蔽
- インタフェースのみ公開



モジュール内で
- どう処理をしているか
- ローカル変数
- ローカルメソッド

を隠蔽する

- モジュール内の変更が他のモジュールに影響しない
- システム全体の見通しが立ちやすい

OSの実現法
抽象データ型(abstract data type)

- 抽象データ型(abstract data type)
  - 複数の手続きによりデータ型を定義

例：時計モジュール

時計モジュール

- 現時刻を教えてくれる
- 時間を計れる
- 決まった時刻にアラーム

OSの実現法
方針と機構の分離

- 方針を決定する部分と実際の処理を行う部分を別のモジュールにする

例：CPUスケジューラとディスパッチャ

方針

CPUスケジューラ
- スケジューリングアルゴリズムの実現
- プロセッサを割り当てるプロセスを選択

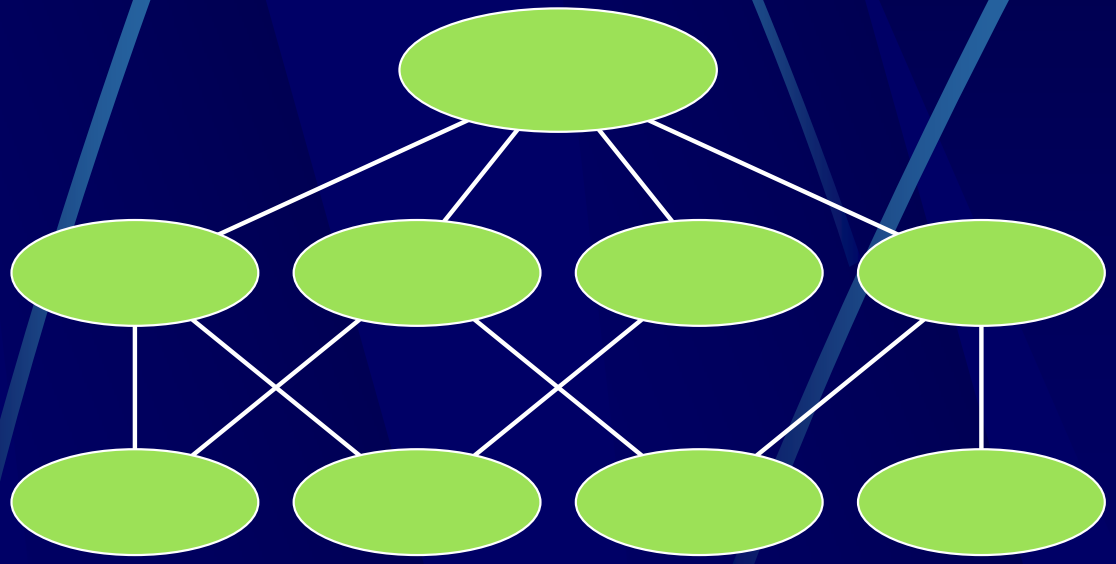
機構

ディスパッチャ
- プロセッサ割り付けの際のレジスタの退避，回復

おのおのの変更，拡張を独立に行える

OSの実現法
階層化(layering)

下位層のサービスを利用
上位層にサービスを提供



メインの手続き

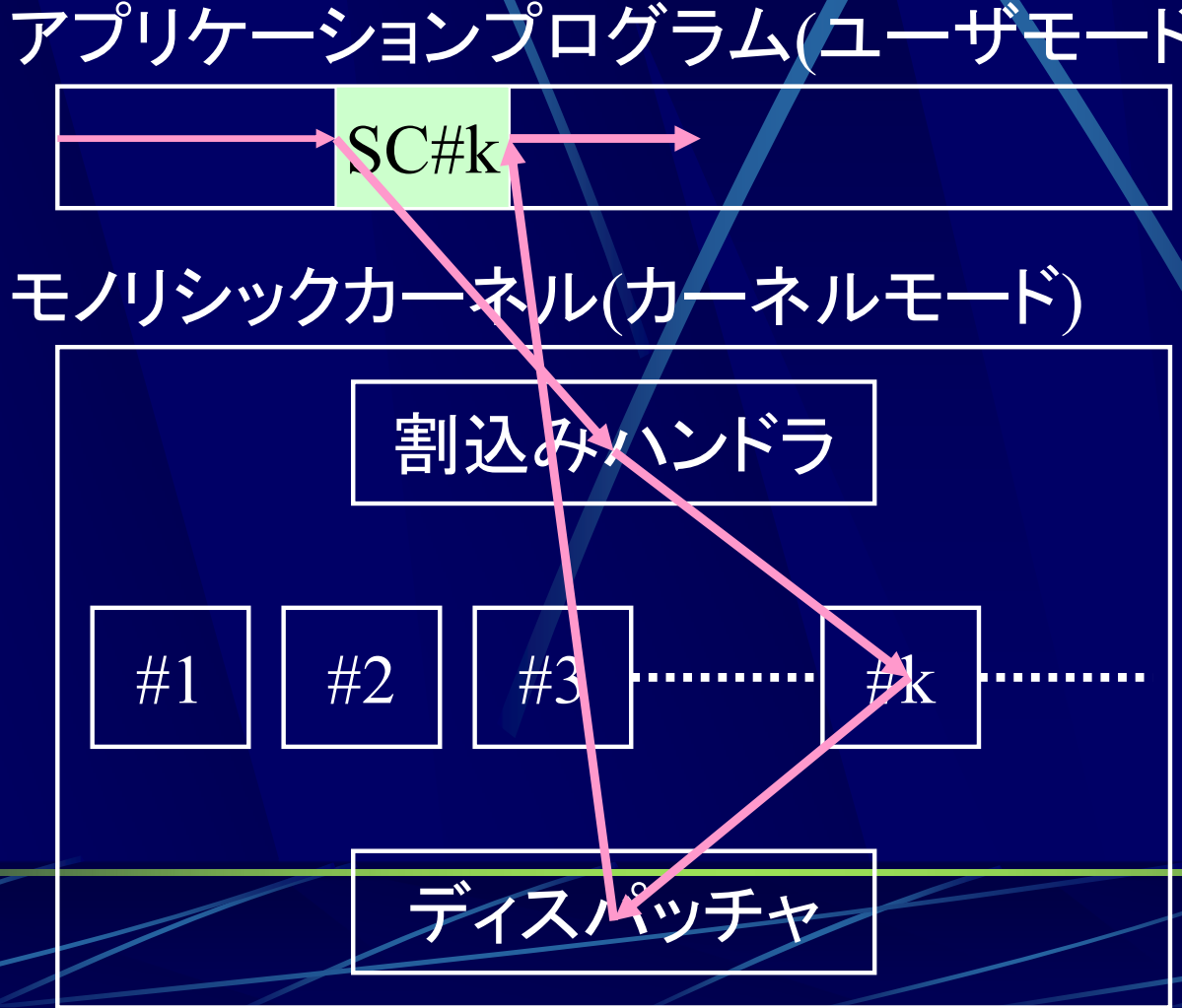
サービス手続き
(システムコールを実行)

ユーティリティ手続き
(サービス手続きの支援)

他の層の実装を知る必要は無い
(インタフェースのみ知っていればよい)
実装を考慮した階層化が必要

OSの実現法
モノリシックカーネル(monolithic kernel)

- OSの機能を全て取り込んだカーネル



monolithic :
一枚岩の, 単層の

OSの実現法
モノリシックカーネル

OSの機能を全て取り込む = 巨大なカーネル

例：UNIX

- 機能強化に伴って巨大化

モノリシックカーネルの問題点

- カーネルはメモリに常駐
  ⇒ 不必要にメモリを占有
- 1つの巨大なプログラム
  ⇒ 機能変更, 拡張に対する柔軟性に欠ける

OSの実現法
マイクロカーネル(micro kernel)

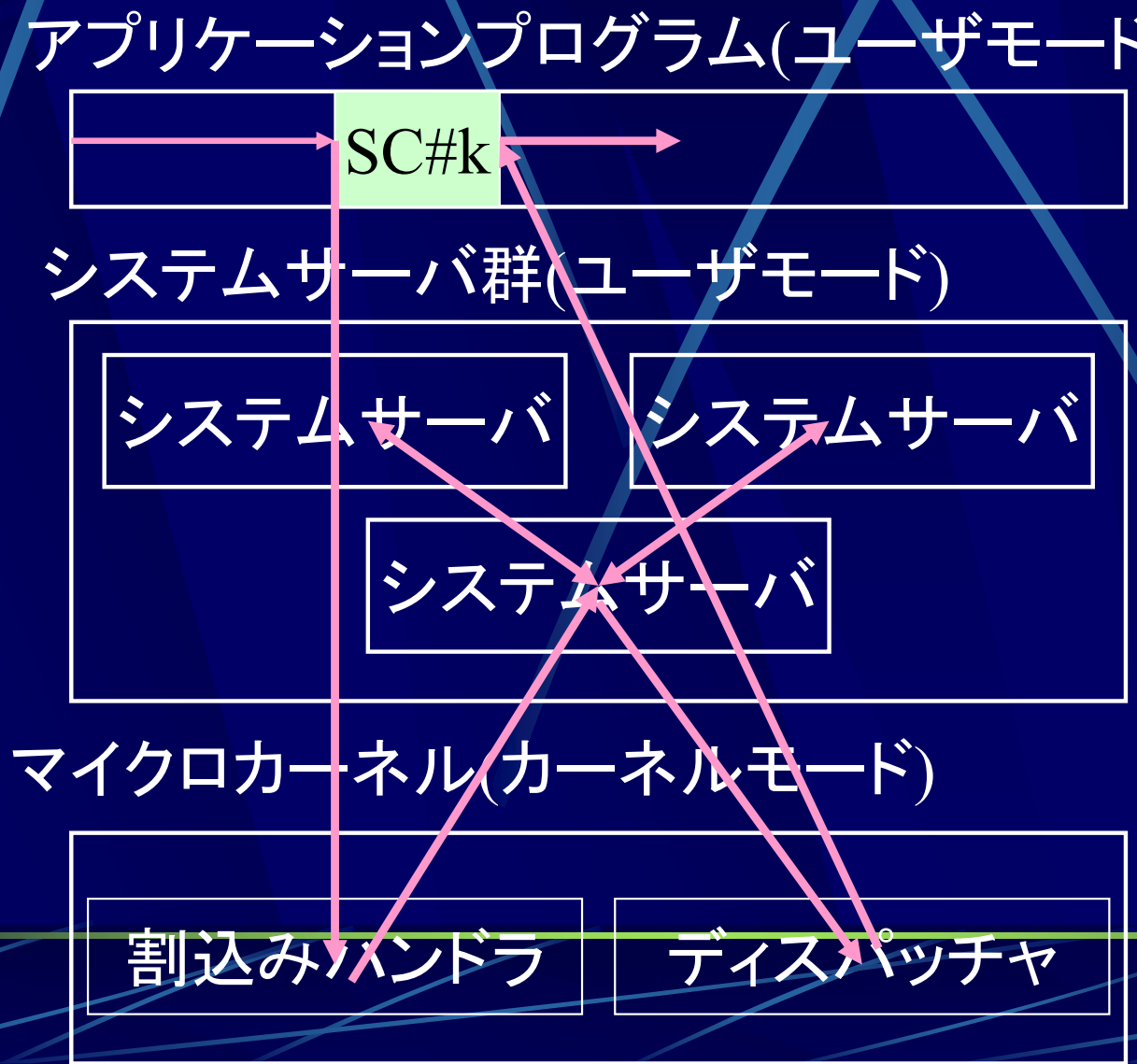
カーネル+システムサーバで構成

- カーネル
  - 割込み処理, スケジューリング, プロセス間通信, メモリ管理等
- システムサーバ (system server)
  - メモリマネージャ, ファイルサーバ, ネームサーバ, デバイスドライバ等

カーネルは必要最低限の機能のみを持つ

クライアント・サーバ型

OSの実現法
マイクロカーネル



OSの実現法
マイクロカーネル

- 長所
  - 見通しの良いシステム設計
  - 変更, 拡張の容易性
  - 分散システムに適合
- 短所
  - プロセス間通信の多用によるオーバヘッド

まとめ:OSの概要

- 多重(マルチ)プログラム
  - 複数のプログラムを見かけ上同時に実行
  - CPUの遊び時間を減らせる
- 割り込み
  - 実行中のプログラムを中断して特別な処理をする
  - カーネル(特権)モードで割り込み処理ルーチン
- システムコール
  - アプリケーションプログラムから OSへのサービス要求

まとめ:カーネルの構成要素

1. 割り込み制御
2. 入出力制御
3. タイマ管理
4. 記憶管理
5. CPUスケジューラ
6. プロセス管理
7. 同期と通信制御
8. ファイルシステム

- カーネルの実現
  - モジュール化
  - 情報隠蔽抽象データ型
  - 階層化
  - モノリシックカーネル
    or マイクロカーネル

課題テスト

- 毎週 GoogleClassroom上で課題テストを行う
  - 授業後～翌週の授業開始まで
- GoogleClassroomで
  オペレーティングシステム
  ⇒授業
  ⇒その回の課題
  と辿る