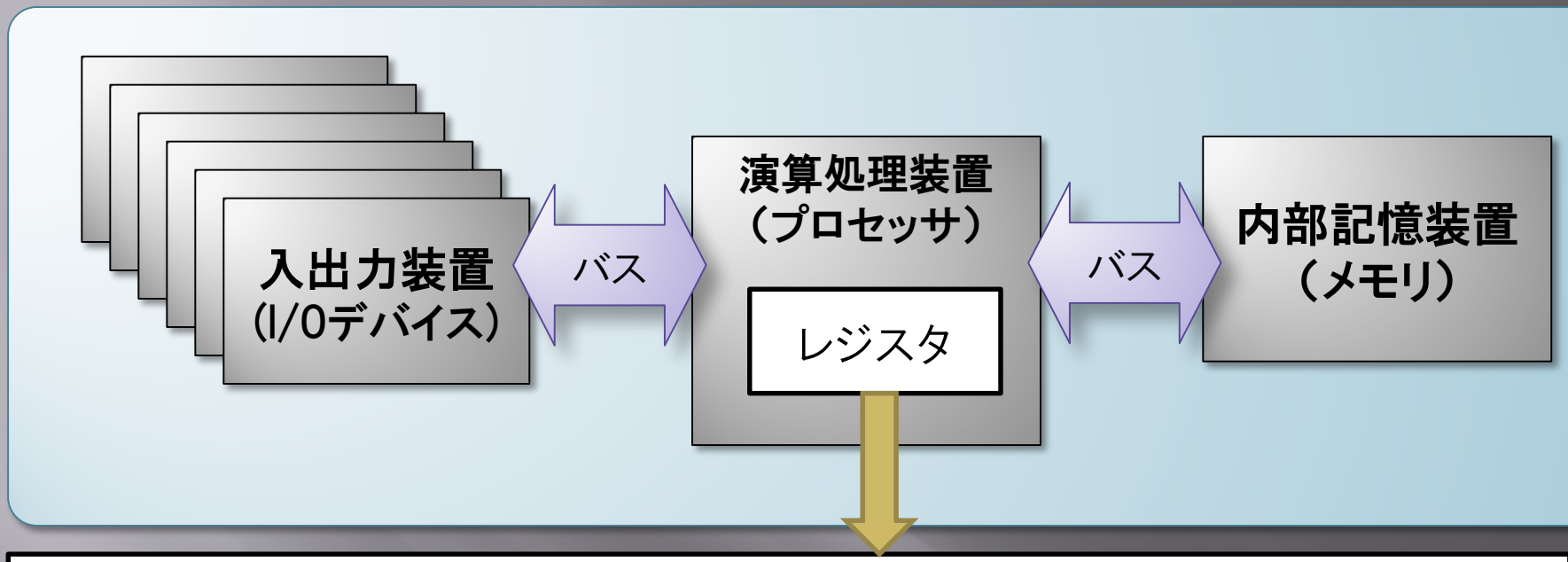
オペレーティングシステム

第3回　実行管理

情報科学メジャー
鏑木崇史



レジスタ

制御レジスタ

プログラムカウンタ

プロセッサステータスレジスタ

汎用レジスタ

プログラムカウンタ （PC）

次に実行する命令のアドレスを示すレジスタ

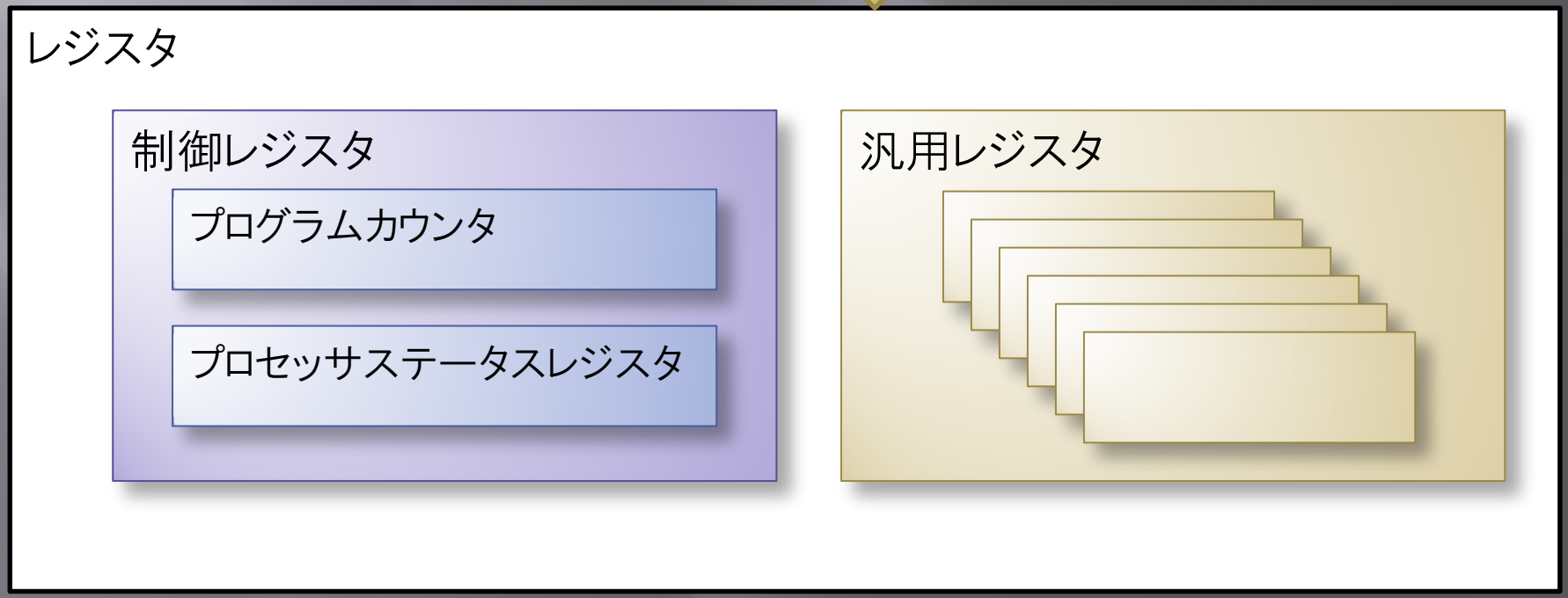
プロセッサ・ステータス・レジスタ （PSW）

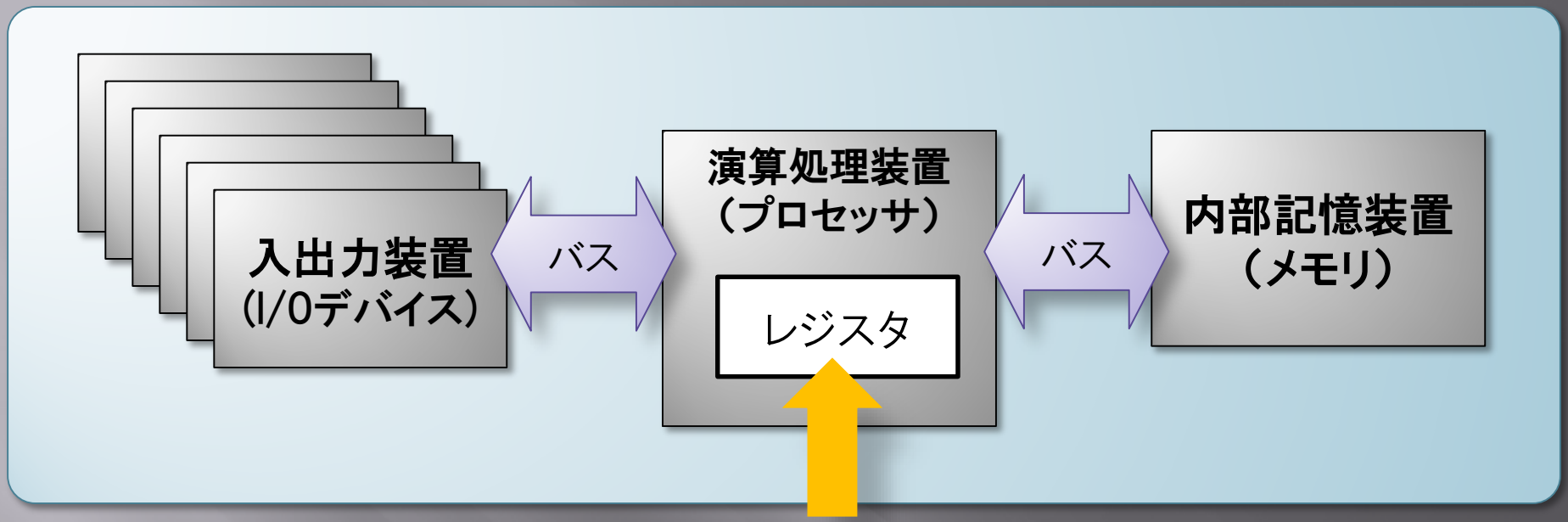
プロセッサの状態を示すフラグを集めたレジスタ
例） キャリー（桁上がり）・オーバーフロー・割込みなど

レジスタ

汎用レジスタ

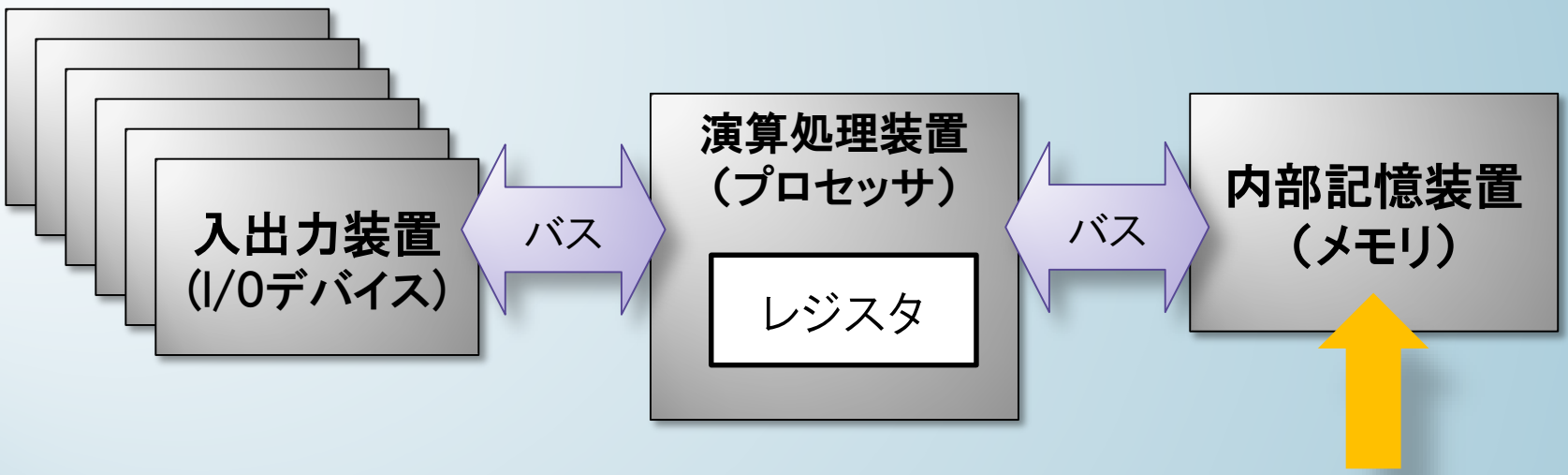
用途が指定されておらず、データを保存し、演算に利用するためのレジスタ





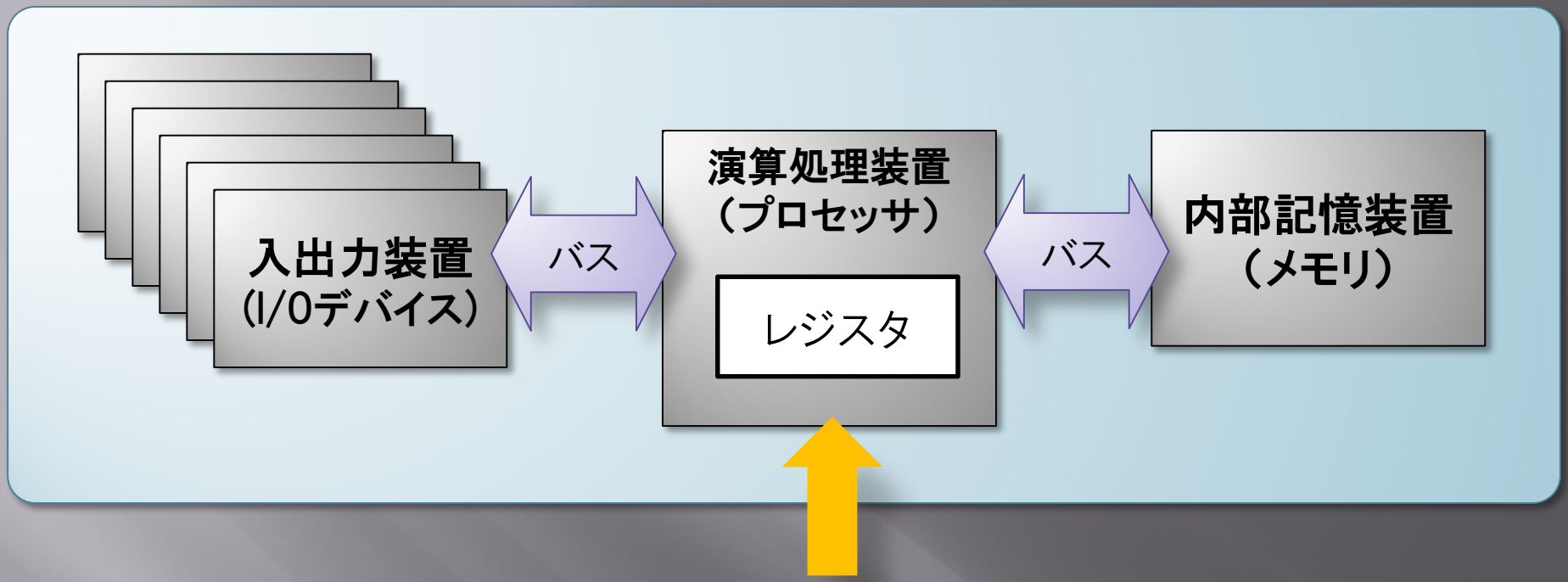
コンテキスト空間

制御レジスタ　＋　汎用レジスタ
→プロセッサ内の記憶空間



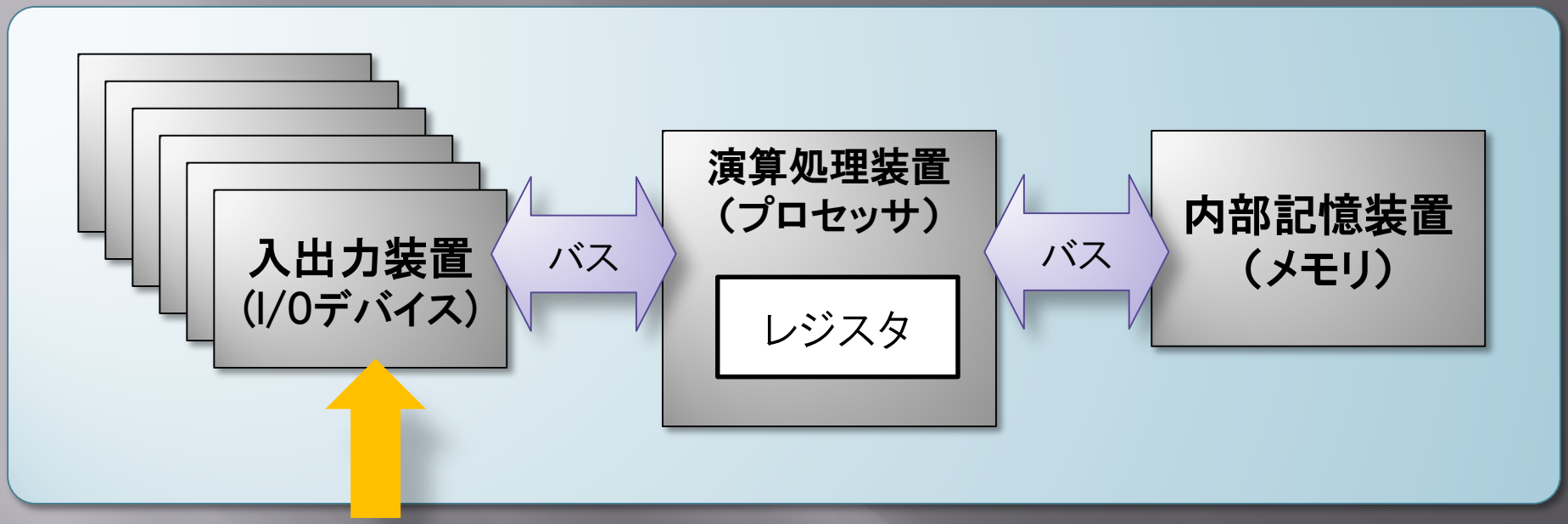
メモリ空間

内部記憶が存在する空間　（アドレス空間とも呼ぶ）



命令セット空間

プロセッサの持つ命令セットによって構成される空間
→一般に固定だが、ヘテロジニアスプロセッサでは異なることもある



I/O空間

I/Oデバイスが配置され、制御や情報交換に利用される空間

コンテキスト空間

制御レジスタ　＋　汎用レジスタ
→プロセッサ内の記憶空間

メモリ空間

内部記憶が存在する空間　（アドレス空間とも呼ぶ）

命令セット空間

プロセッサの持つ命令セットによって構成される空間
→一般に固定だが、ヘテロジニアスプロセッサでは異なることもある

I/O空間

I/Oデバイスが配置され、制御や情報交換に利用される空間

タスク

プロセッサで実行されるプログラム本体

スレッド

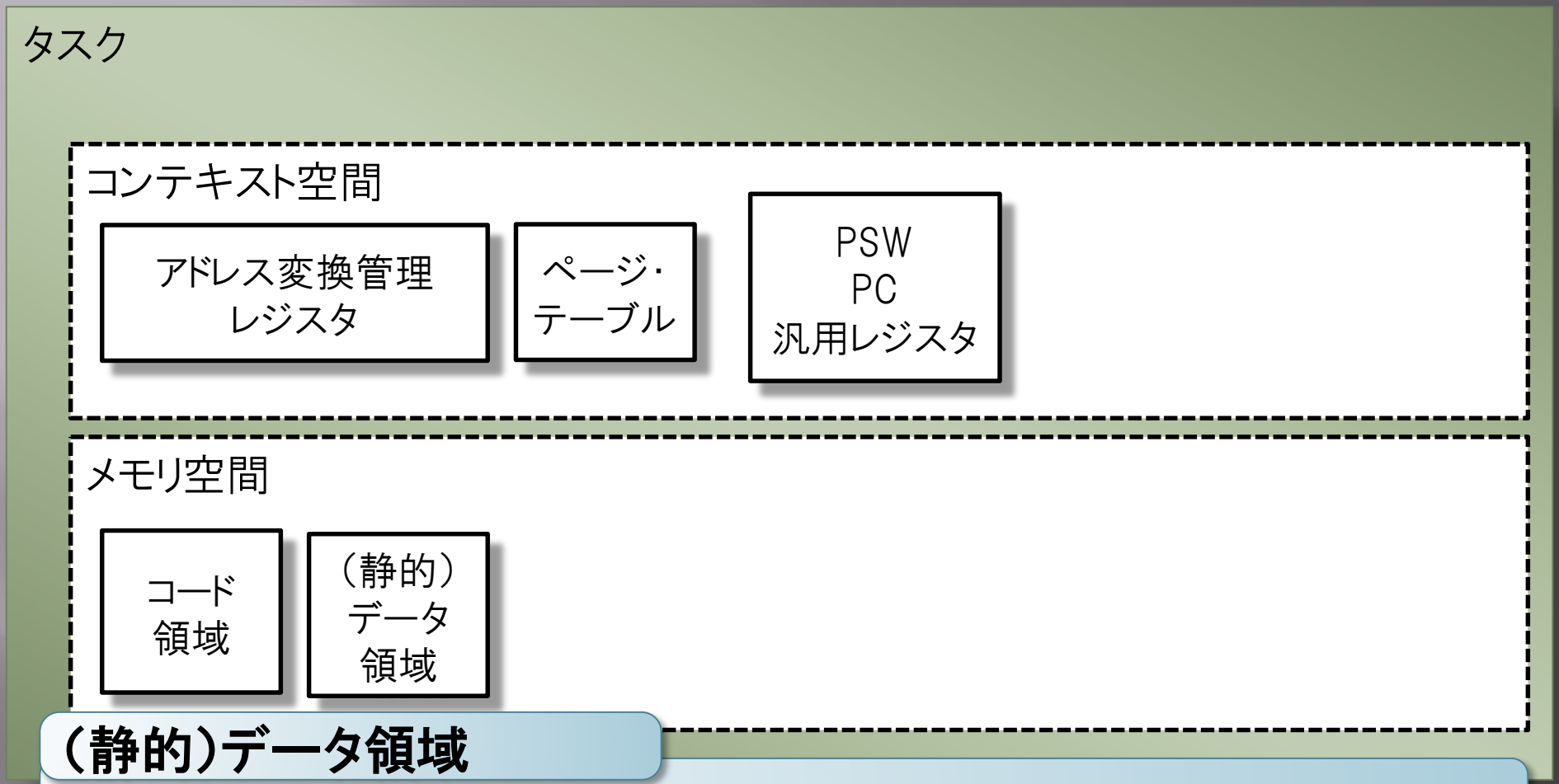
タスクの中には1つ以上のスレッドが存在する
同じ **メモリ空間と命令セット空間** を共有するが
異なる **コンテキスト空間** を有する
軽量プロセスとも呼ばれる

実行主体

コード領域

プログラム本体が保存されている領域

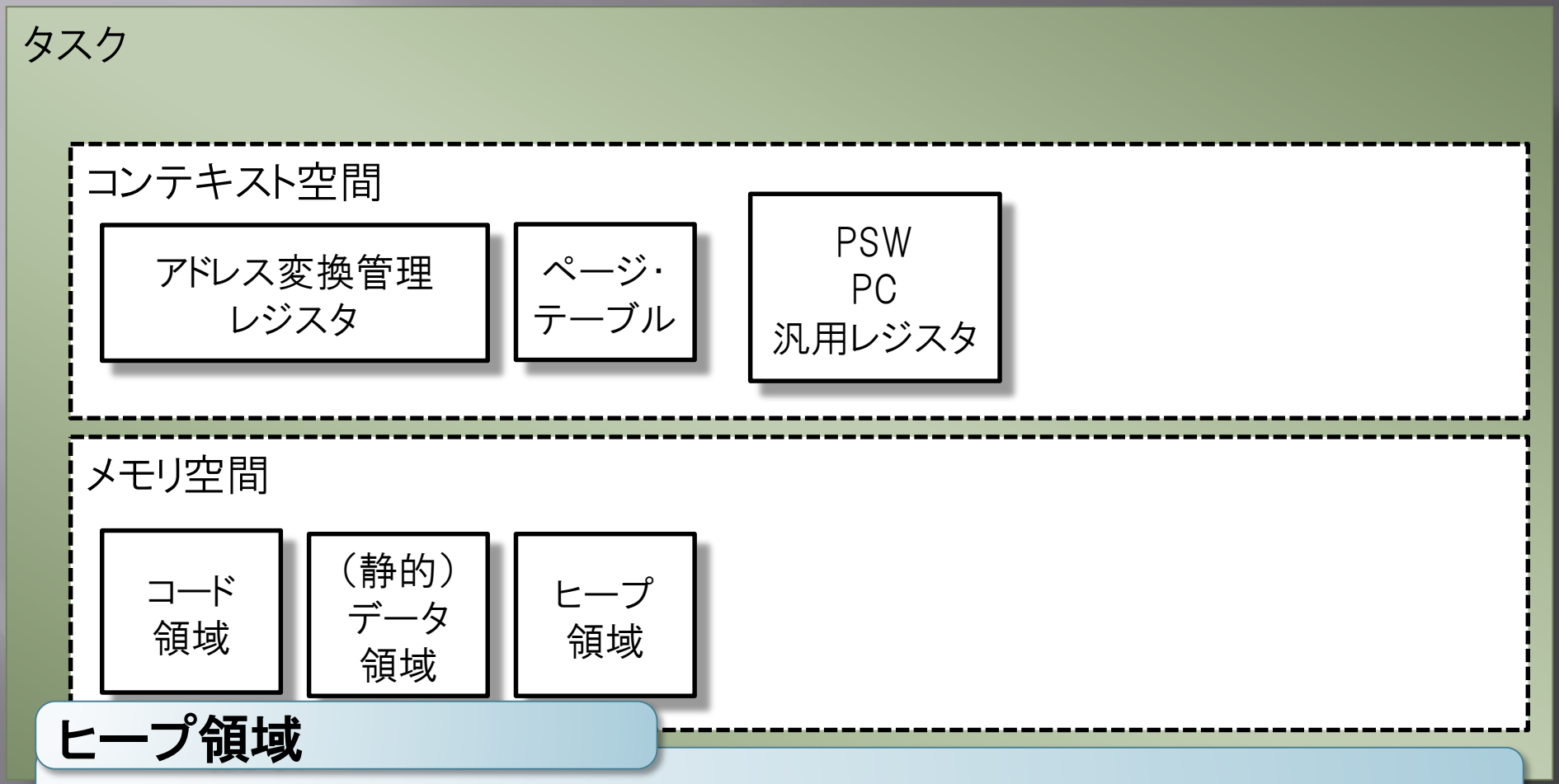
実行主体



（静的）データ領域

プログラム実行の最初から存在し、プログラム全体で利用される
→ C言語のグローバル変数やstatic変数

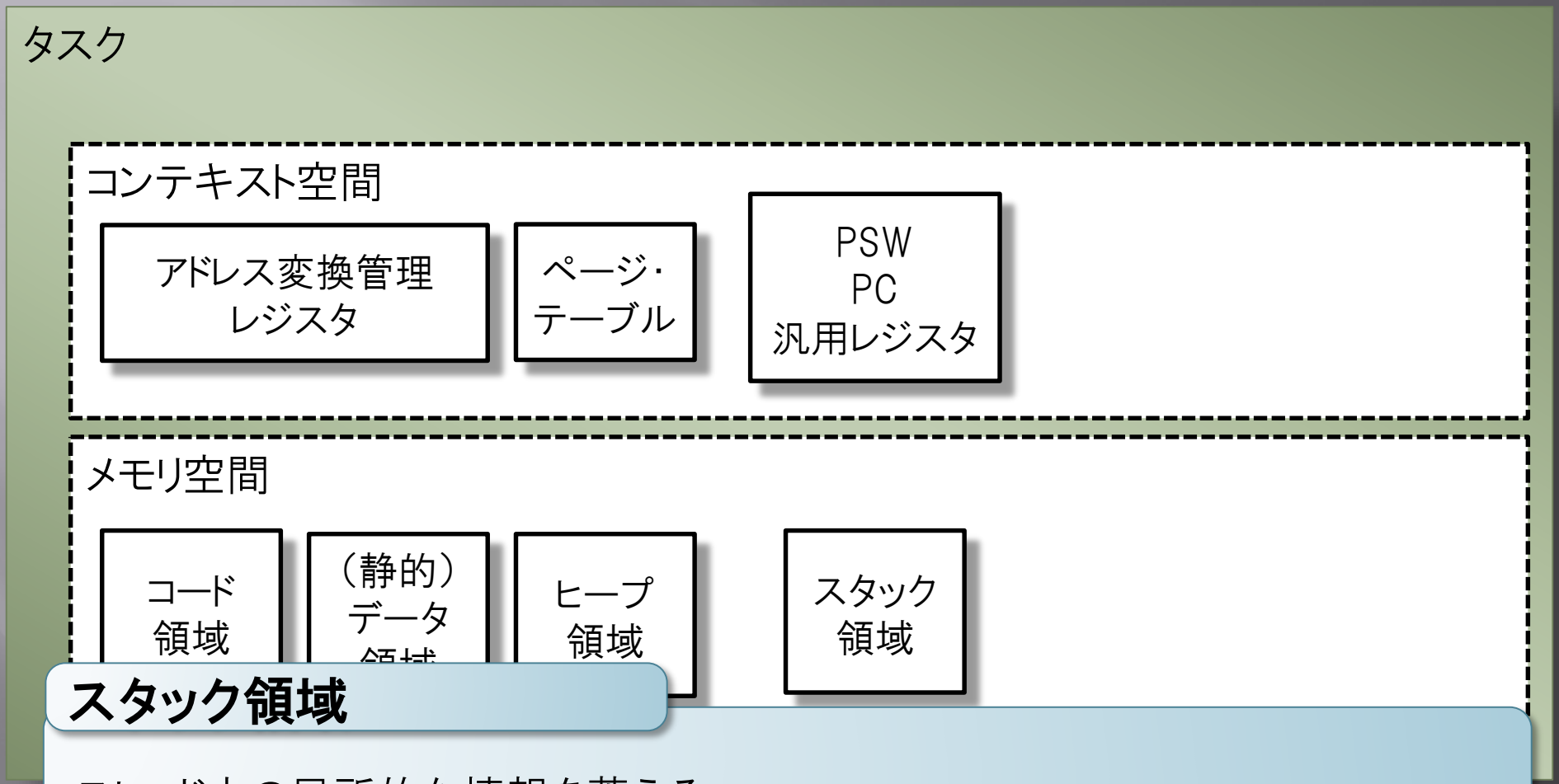
実行主体



ヒープ領域

プログラムの実行に伴って動的に割り当てられる
→ C言語のmallocやfreeなどで利用できる領域

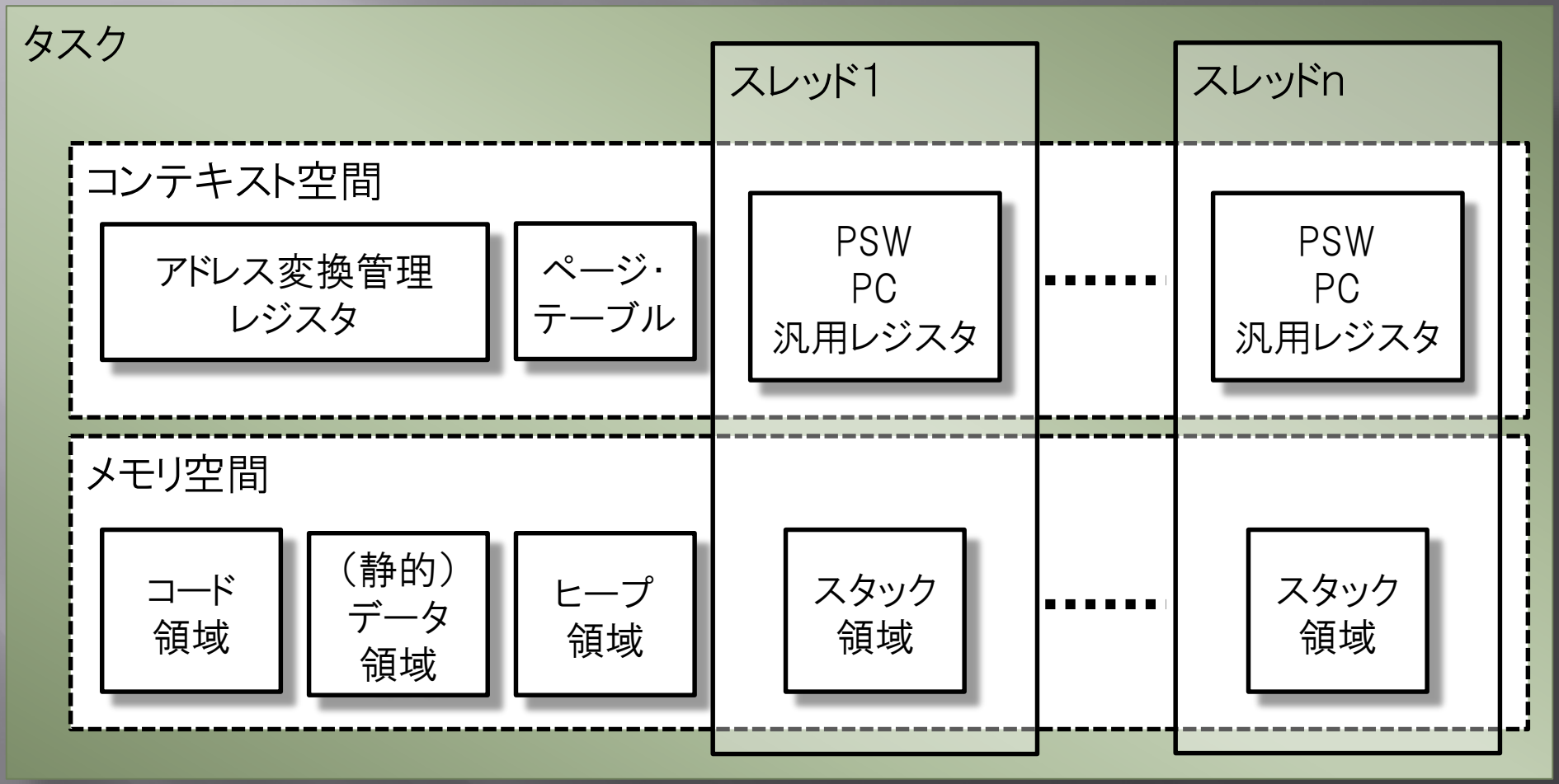
実行主体



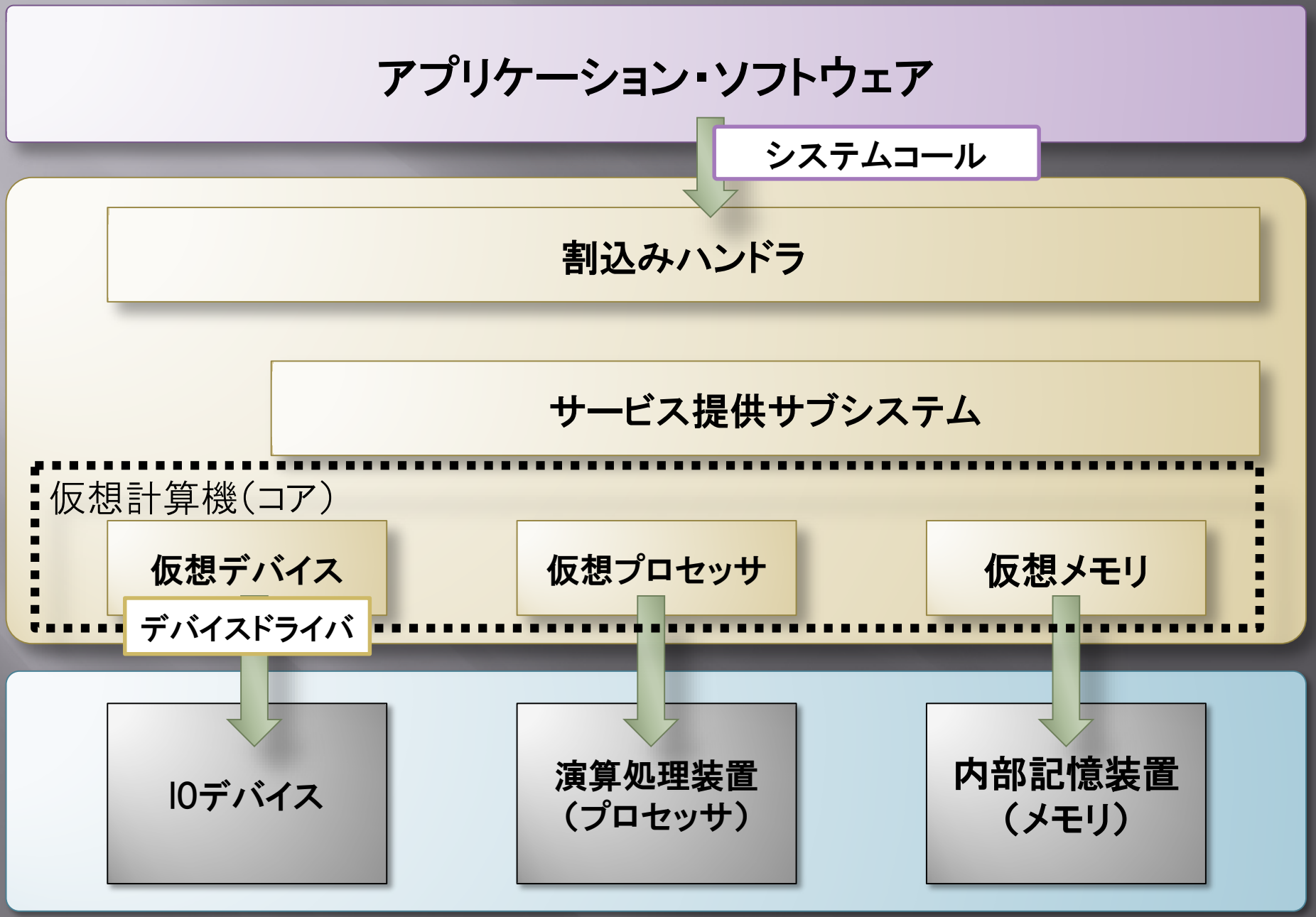
スタック領域

スレッド内の局所的な情報を蓄える
コンパイラやOSが自動で割当・解放を行う
サイズはコンパイル・リンク時に決定される

実行主体


タスク
スレッド1
スレッドn
コンテキスト空間
アドレス変換管理
レジスタ
ページ・
テーブル
PSW
PC
汎用レジスタ
PSW
PC
汎用レジスタ
メモリ空間
コード
領域
（静的）
データ
領域
ヒープ
領域
スタック
領域
スタック
領域

オペレーティングシステムの構造



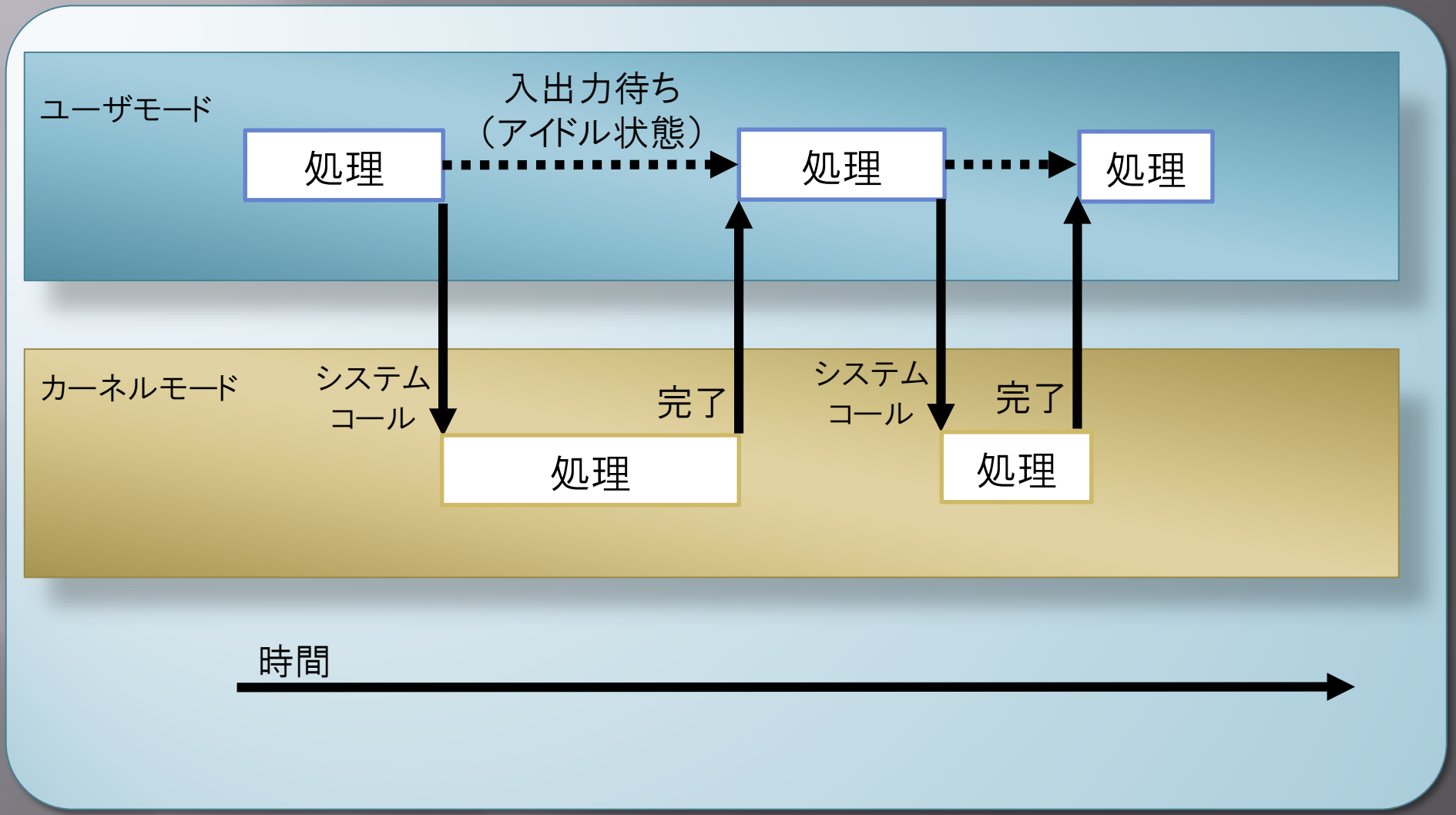
実行モード

特権モード

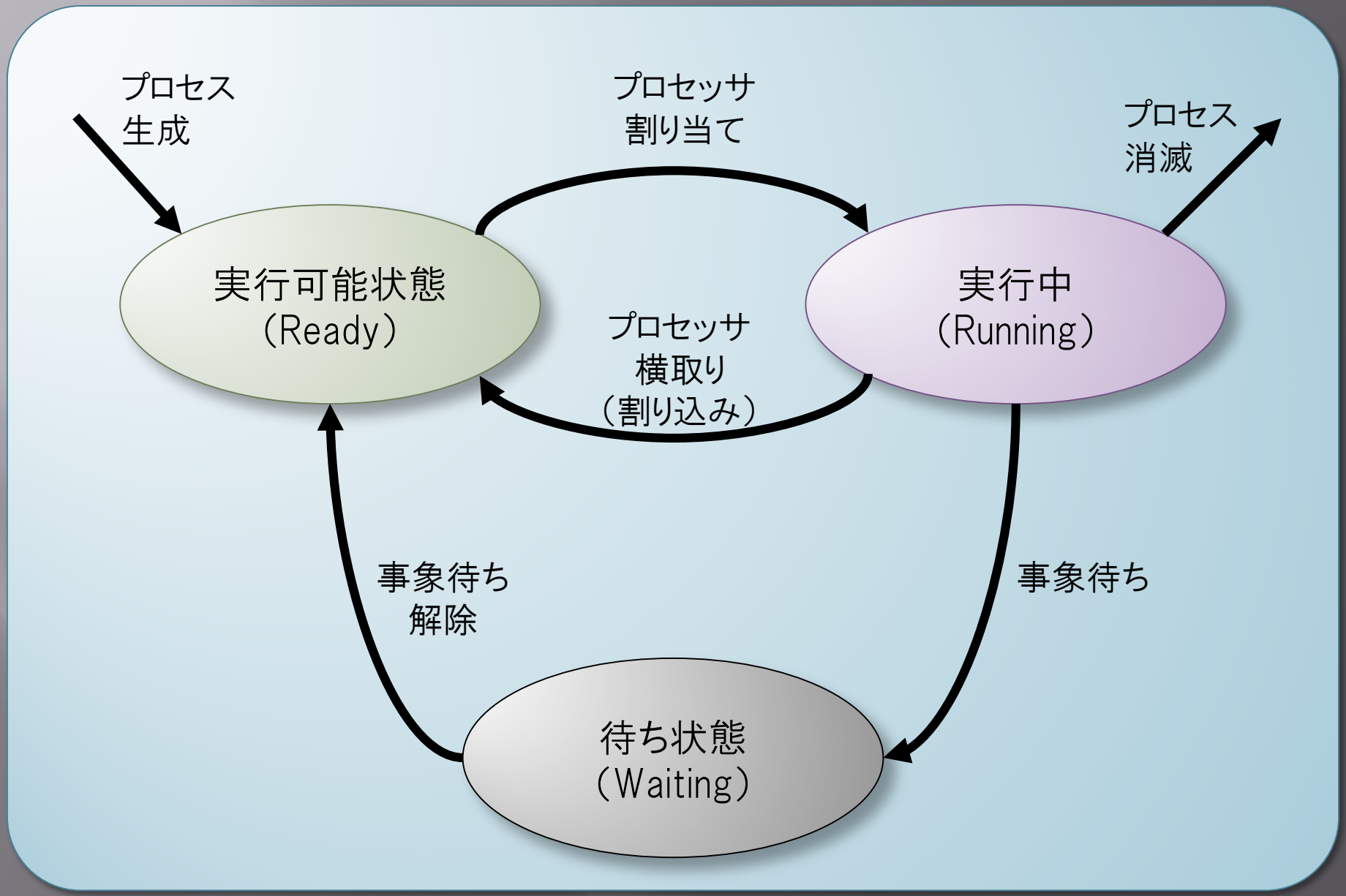
すべての資源へのアクセスが可能
カーネルモードとも呼ばれる

非特権モード

入出力命令、メモリ管理命令、割込み制御命令などの特権命令は実行不可
ユーザモードとも呼ばれる


ユーザモード
入出力待ち
（アイドル状態）
処理
処理
処理
カーネルモード
システム
コール
完了
システム
コール
完了
処理
処理
時間

実行状態


プロセス
生成
プロセッサ
割り当て
プロセス
消滅
実行可能状態
(Ready)
実行中
(Running)
プロセッサ
横取り
(割り込み)
事象待ち
解除
事象待ち
待ち状態
(Waiting)

課題1

以下のプログラムを作成し、関数putsが発行しているシステムコールを観察する

```
#include <stdio.h>

int main(void){
        puts(“Hello world”);
        return 0;
}
```

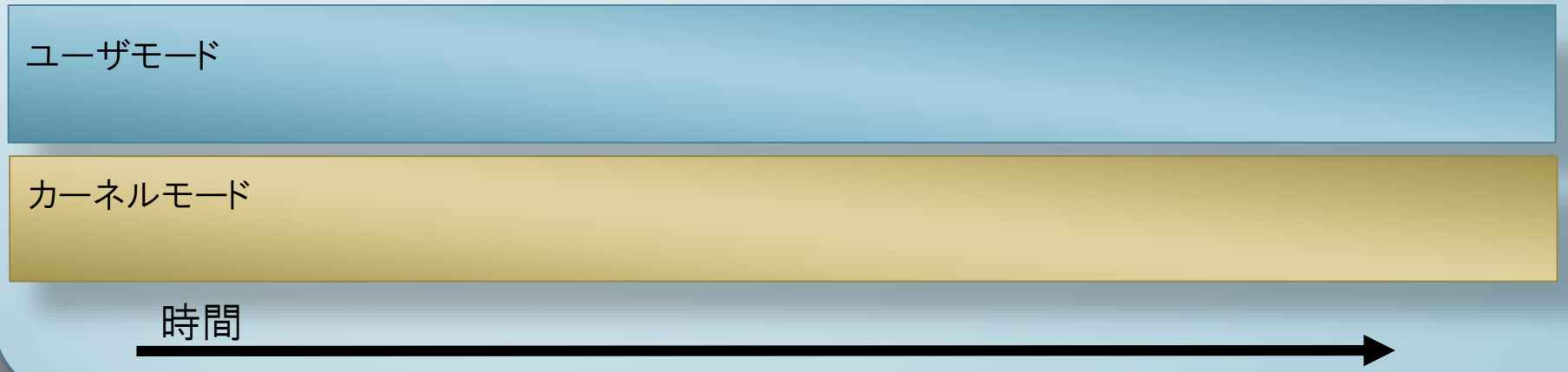
1. EC2を起動し、SSHで接続する
2. vi hello.c と入力
3. i と入力すると入力モードになる
4. プログラムを記述する
5. 終了は「ESC」を押した後、「wq」と入力してEnterキーを押す
6. コンパイルは「cc -o hello hello.c」とする
7. 実行は「./hello」と入力する
8. プログラム中のシステムコールの履歴を保存するには
   「strace -o hello.log ./hello」と入力する
9. システムコールの履歴を閲覧するには
   「less hello.log」と入力する

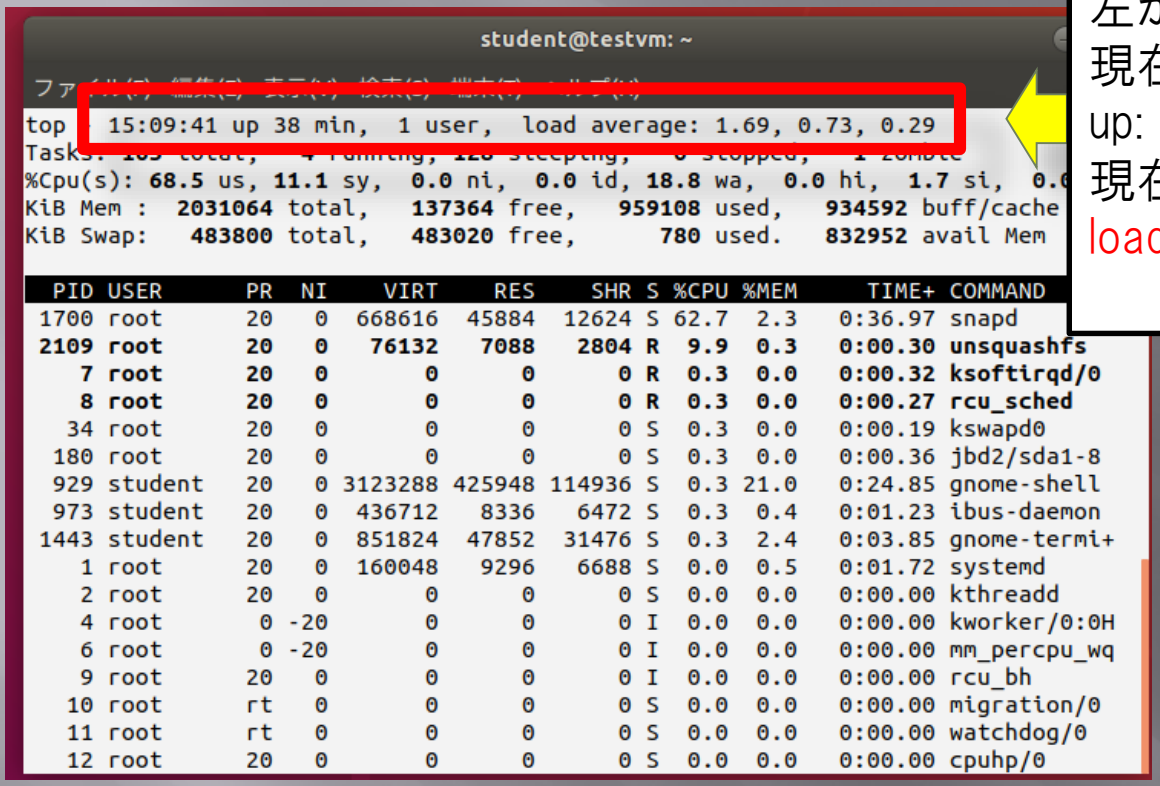
上記の中でwriteがputsの発行したシステムコールである
そのほかのシステムコールはプログラムの開始・終了に伴う処理に必要なもの

課題2

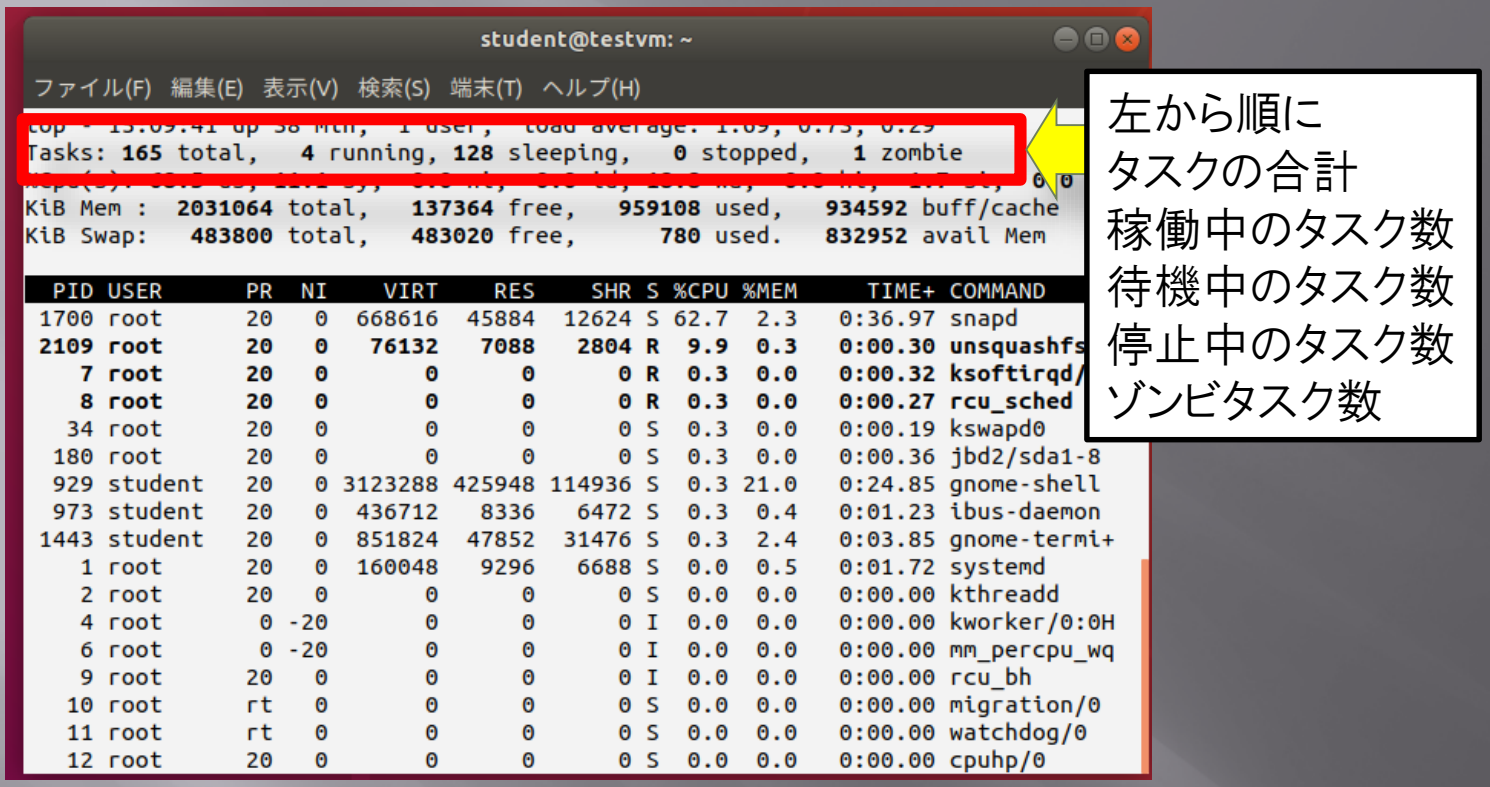
以下のシステムコールを発行しないプログラムを作成し、
ユーザモードとカーネルモードの実行時間の比率を求め、
下記の図に加筆して説明せよ

```
int main(void){
        for(;;)
                ;
        return 0;
}
```

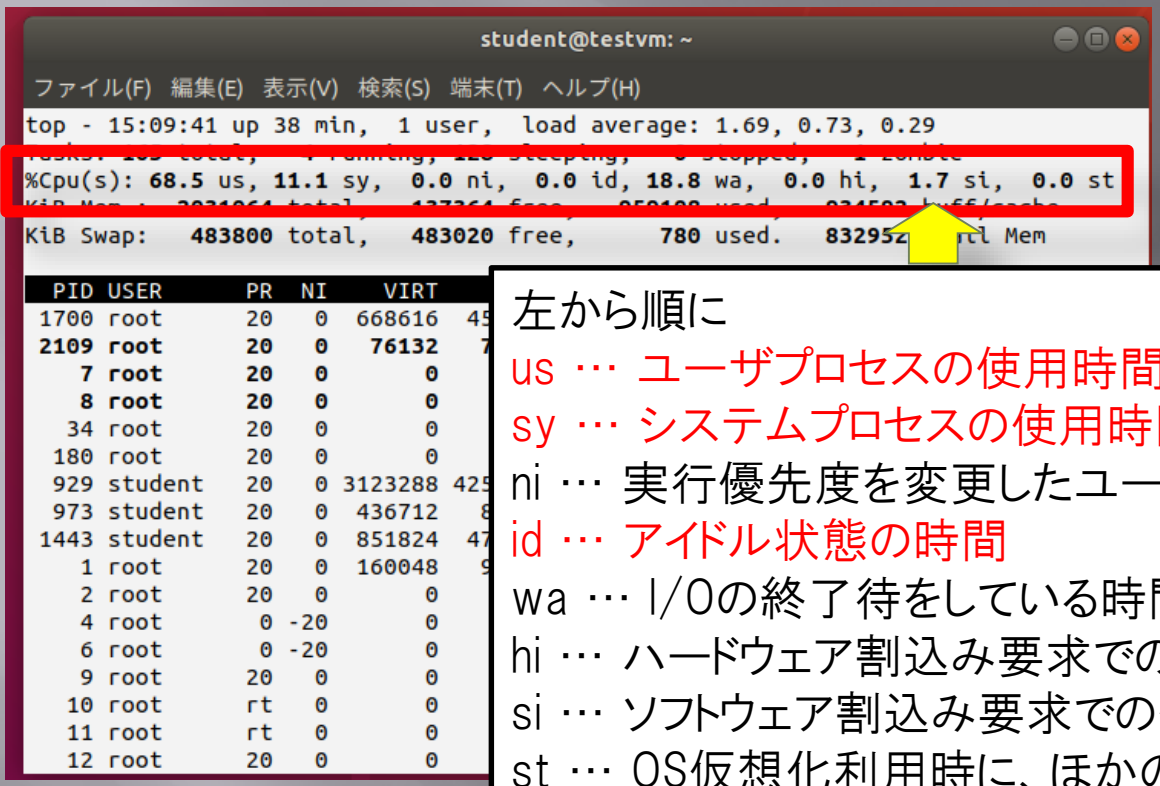




左から順に
現在時刻
up: 稼働時間
現在のログインユーザ数
load average: 時間あたりの待機タスク数
（1分間、5分間、15分間）



左から順に
タスクの合計
稼働中のタスク数
待機中のタスク数
停止中のタスク数
ゾンビタスク数



左から順に

us … ユーザプロセスの使用時間

sy … システムプロセスの使用時間

ni … 実行優先度を変更したユーザプロセスの使用時間

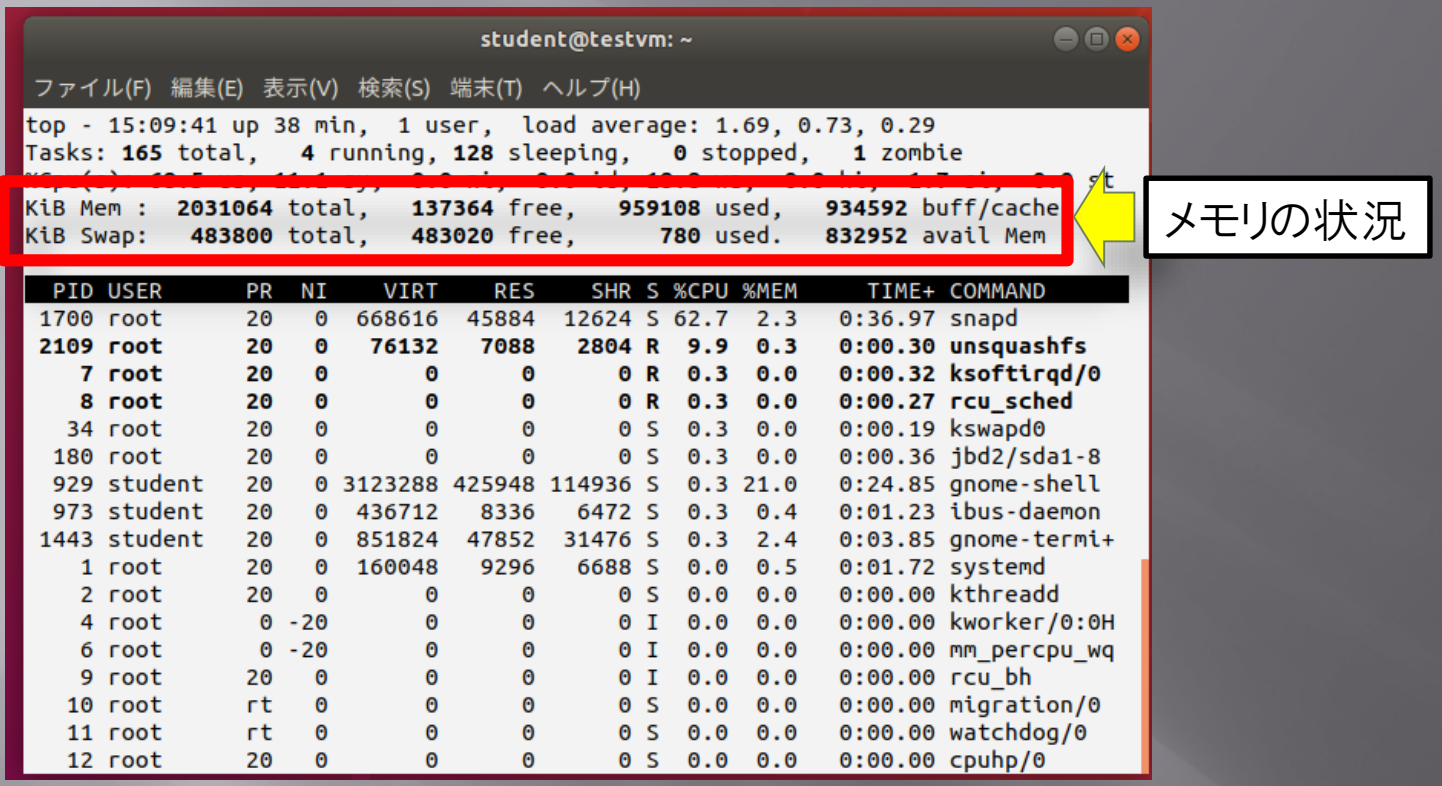
id … アイドル状態の時間

wa … I/Oの終了待をしている時間

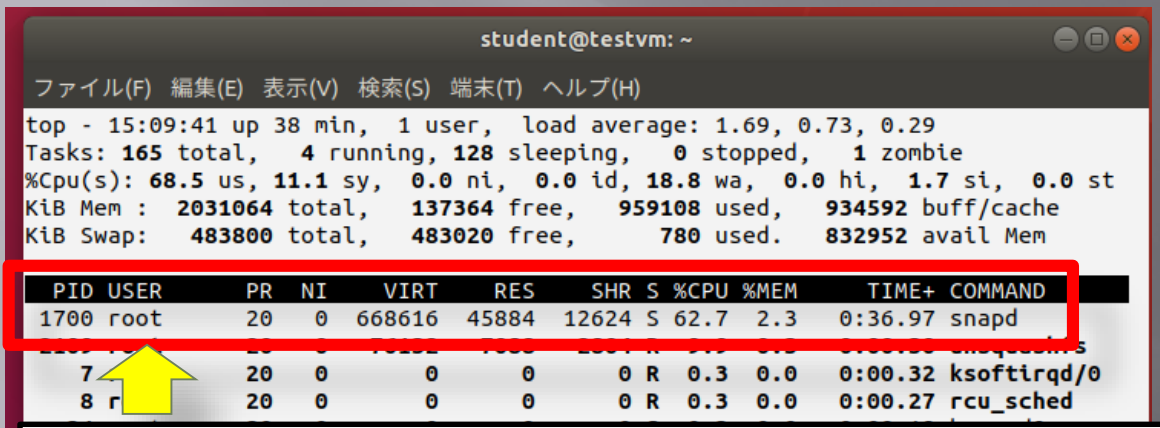
hi … ハードウェア割込み要求での使用時間

si … ソフトウェア割込み要求での使用時間

st … OS仮想化利用時に、ほかの仮想CPUの計算で待たされた時間


student@testvm: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
top - 15:09:41 up 38 min, 1 user, load average: 1.69, 0.73, 0.29
Tasks: 165 total, 4 running, 128 sleeping, 0 stopped, 1 zombie
KiB Mem : 2031064 total, 137364 free, 959108 used, 934592 buff/cache
KiB Swap: 483800 total, 483020 free, 780 used. 832952 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1700 root 20 0 668616 45884 12624 S 62.7 2.3 0:36.97 snapd
2109 root 20 0 76132 7088 2804 R 9.9 0.3 0:00.30 unsquashfs
7 root 20 0 0 0 0 R 0.3 0.0 0:00.32 ksoftirqd/0
8 root 20 0 0 0 0 R 0.3 0.0 0:00.27 rcu_sched
34 root 20 0 0 0 0 S 0.3 0.0 0:00.19 kswapd0
180 root 20 0 0 0 0 S 0.3 0.0 0:00.36 jbd2/sda1-8
929 student 20 0 3123288 425948 114936 S 0.3 21.0 0:24.85 gnome-shell
973 student 20 0 436712 8336 6472 S 0.3 0.4 0:01.23 ibus-daemon
1443 student 20 0 851824 47852 31476 S 0.3 2.4 0:03.85 gnome-termi+
1 root 20 0 160048 9296 6688 S 0.0 0.5 0:01.72 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
9 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rcu_bh
10 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
11 root rt 0 0 0 0 S 0.0 0.0 0:00.00 watchdog/0
12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0

メモリの状況



左から順番に

PID … プロセスID

USER … プロセスの実行ユーザ

PR … プロセスの静的優先度(値が低い方が優先度が高い)

NI … プロセスの相対優先度(0を基準とし、-20(優先度高) ～ 19(優先度低)で表している)

VIRT … プロセスの仮想メモリーサイズ(スワップアウトしたメモリー使用量を加えたメモリー容量)

RES … プロセスの使用しているメモリー容量(物理メモリー容量)

SHR … プロセスの使用している共有メモリー容量

S … プロセスの状態

%CPU … CPU使用率

%MEM … 物理メモリー使用率

TIME+ … プロセスの実行時間

COMMAND … プロセスで実行されているコマンド

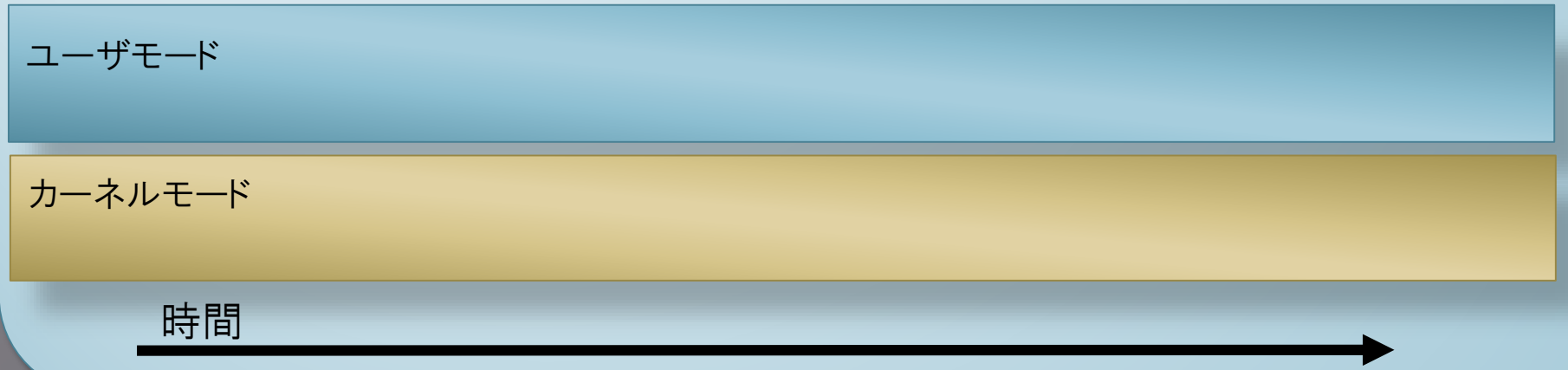
1. vi loop.c と入力
2. i と入力すると入力モードになる
3. プログラムを記述する
4. 終了は「ESC」を押した後、「wq」と入力してEnterキーを押す
5. コンパイルは「cc -o loop loop.c」とする
6. 実行は「./loop &」と入力する（直後に表示される数字がプロセス番号）
7. 実行中の処理時間を閲覧するためには「top」と入力する　終了は「q」
8. 実行したプロセスを終了するには
   「kill -KILL <プロセス番号>」と入力する

課題3

以下のシステムコールを繰り返し発行するプログラムを作成し、
ユーザモードとカーネルモードの実行時間の比率を求め、
下記の図に加筆して説明せよ

```
#include <sys/types.h>
#include <unistd.h>

int main(void){
        for(;;)
                getppid();
        return 0;
}
```



1. vi ppidloop.c と入力
2. i と入力すると入力モードになる
3. プログラムを記述する
4. 終了は「ESC」を押した後、「wq」と入力してEnterキーを押す
5. コンパイルは「cc -o ppidloop ppidloop.c」とする
6. 実行は「./ppidloop &」と入力する
7. 実行中の処理時間を閲覧するためには「top」と入力する　終了は「q」
8. 実行したプロセスを終了するには
   「kill -KILL <プロセス番号>」と入力する

9. 最後にEC2を終了する

連絡先

鏑木　崇史
E-mail: kabutakashi@icu.ac.jp