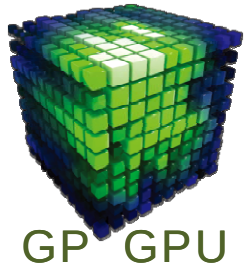
Introduction to GPU Parallel Computing

東京工業大学　学術国際センター
青木　尊之

並列処理



GPUの内部には400以上のプロセッサ（CUDAコア）がある。これをうまく使うことによりGPU本来の性能を引き出すことができる。

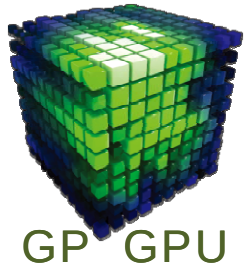
並列処理（計算）を制する（マスターする）者は、GPUコンピューティングを制す。

並列計算は、

タスク並列　と　データ並列

に分けられる。

例：夏休みの宿題の分担

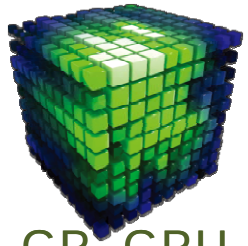


仲の良い4人組（青木君、斉藤君、佐藤君、鈴木君）は、夏休みの宿題を楽に済ませようと考え、分担して片付けることにした。

・数学	大問1つの中に小問が3題
・英語	大問1つの中に小問が4題
・物理	大問1つの中に小問が5題
・化学	大問1つの中に小問が4題

青木君が数学、斉藤君が英語、佐藤君が物理、鈴木君が化学を担当する。

数学の中間テスト(1)

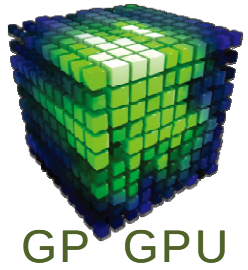


生徒48人のクラスで数学の中間テストを行う。カンニングを嫌う先生が、48ページの問題集の中から、生徒がそれぞれ違うページの問題を解いて、先生に提出することにする。その問題集は、数値が違うがほぼ同じような問題が続いている。

先生が行うこと:

- 事前に問題集の配布。
- クラスの生徒を班に分ける。
- 問題の解き方の解説プリントの配布。
- 採点と平均点の算出。

数学の中間テスト(2)

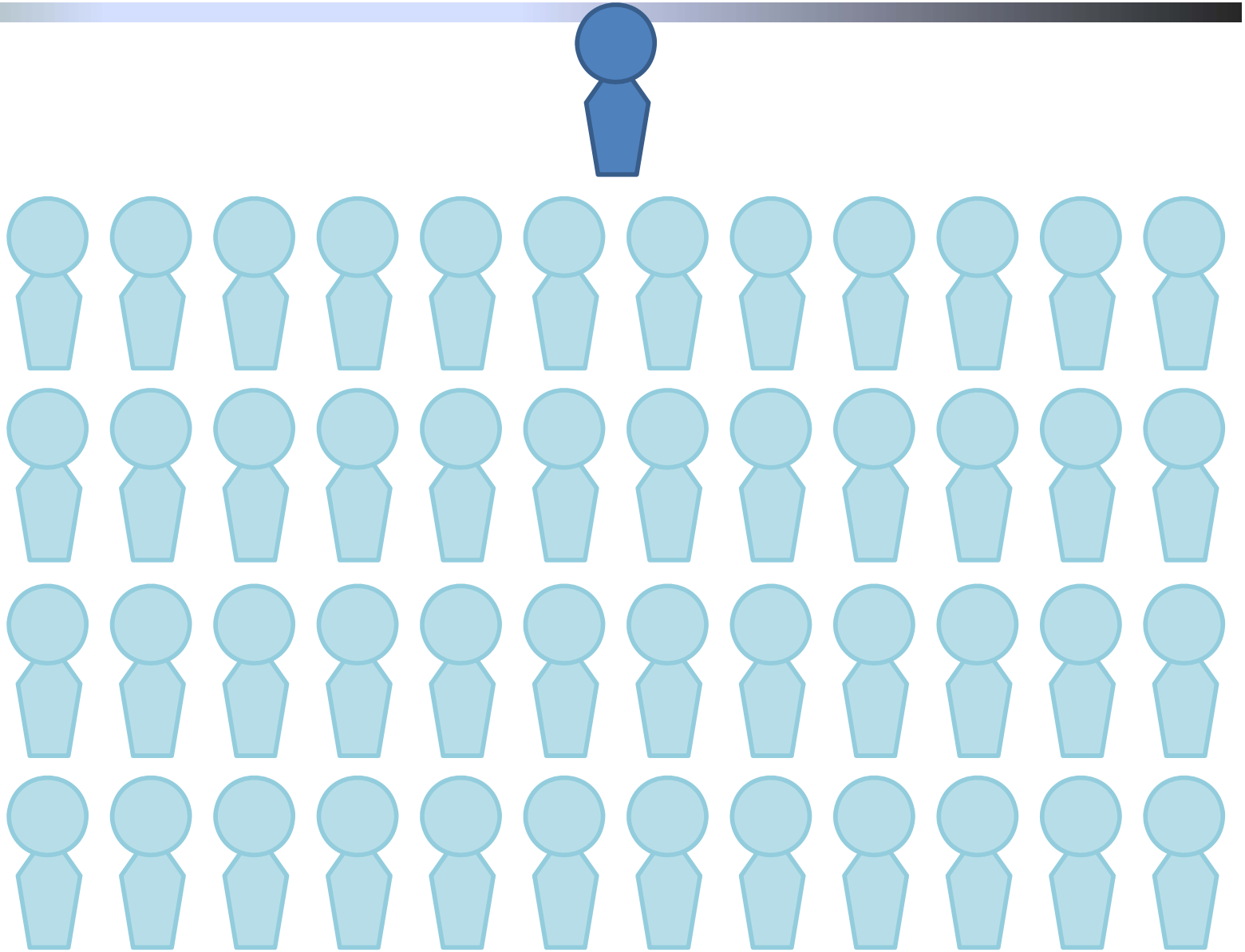


生徒が行うこと:

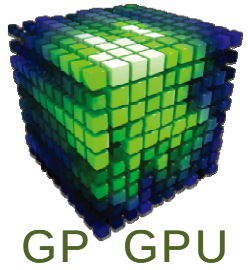
- 自分はどの問題をやればよいか指示を聞く。
- 問題を解いて計算する。
- 何ページの問題を解いたかと、その答えを解答用紙に記入し、名前を書いて提出。

教師はクラス全員に同じ指示しか出せない。

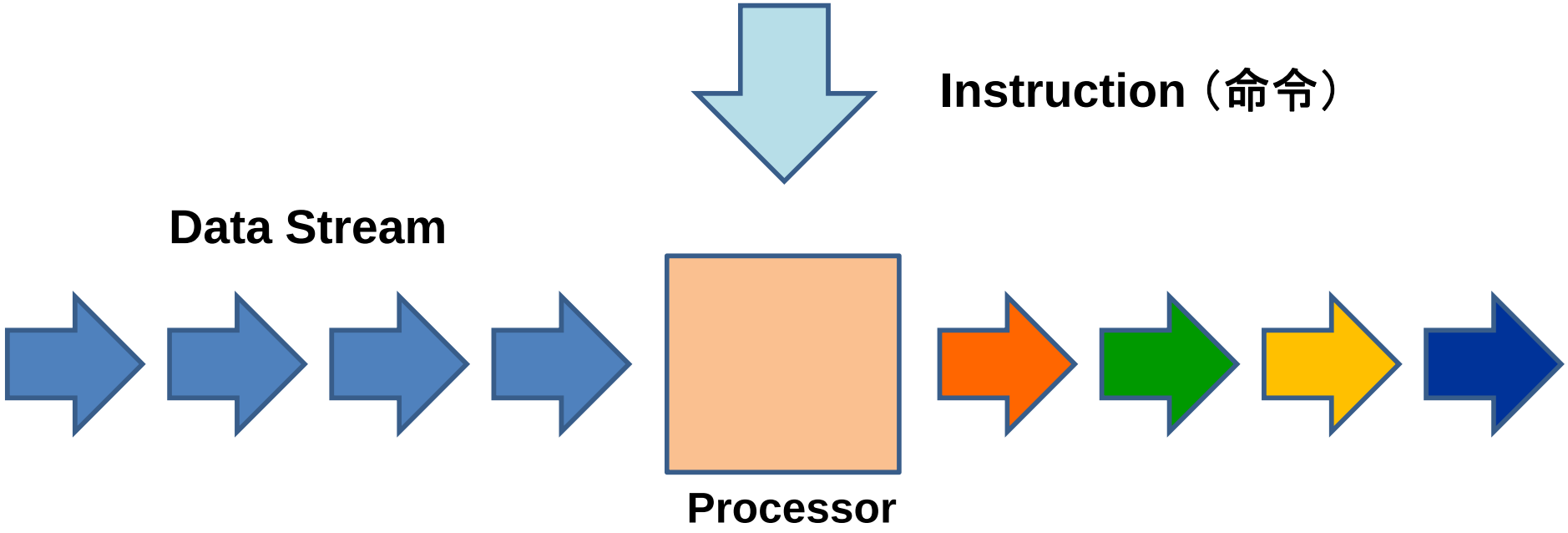
クラスの班分け



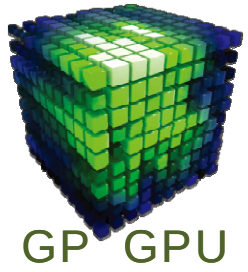
並列計算のアーキテクチャ



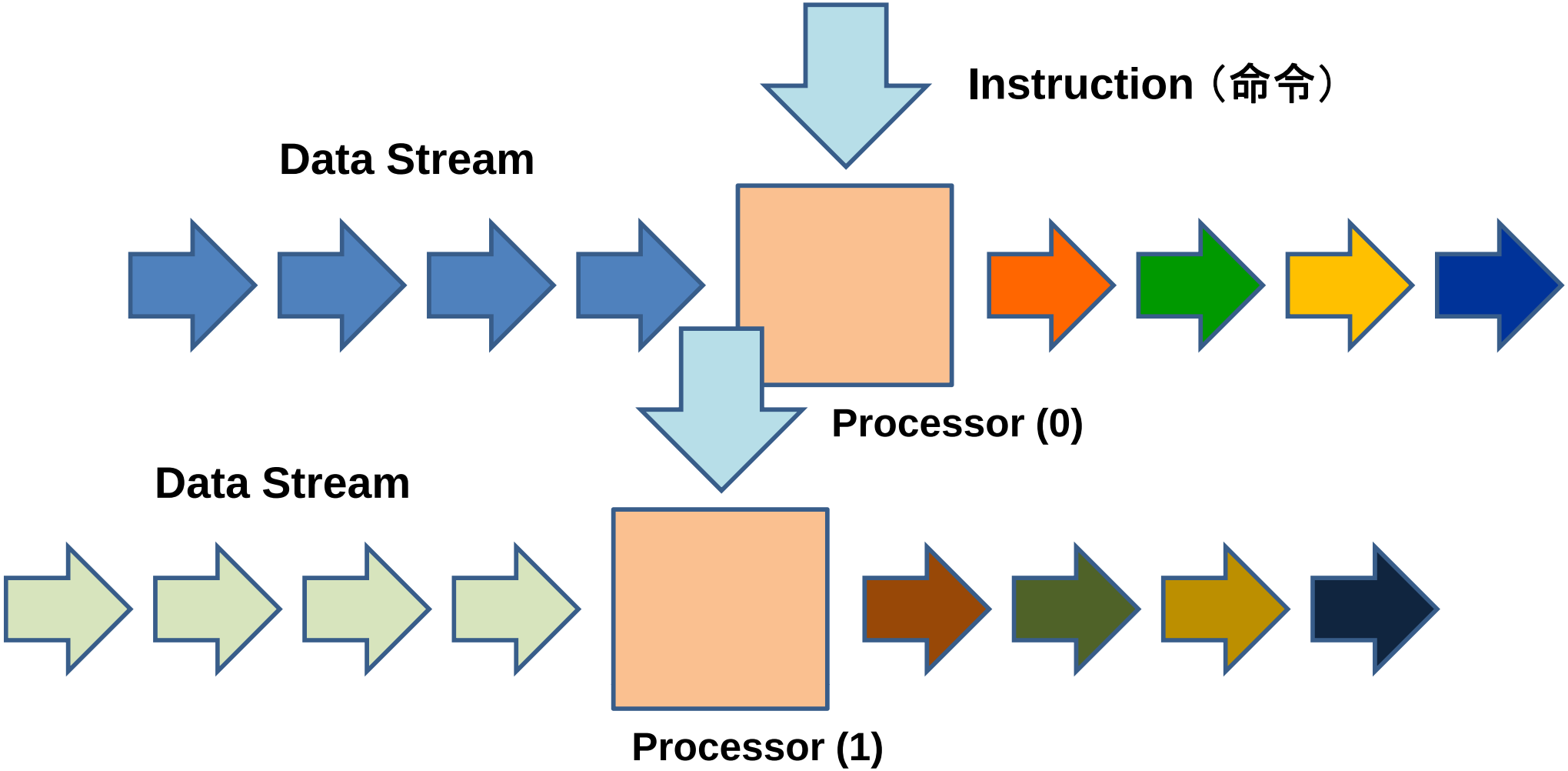
SISD: Single Instruction Single Data Stream



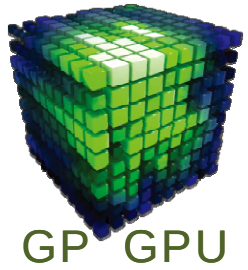
並列計算のアーキテクチャ



SIMD: Single Instruction Multiple Data Stream



並列計算のアーキテクチャ



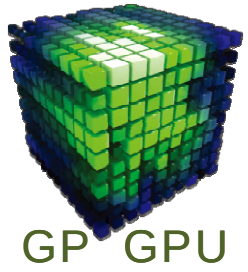
MISD: Multiple Instruction Single Data Stream

（ほとんど存在しない。）

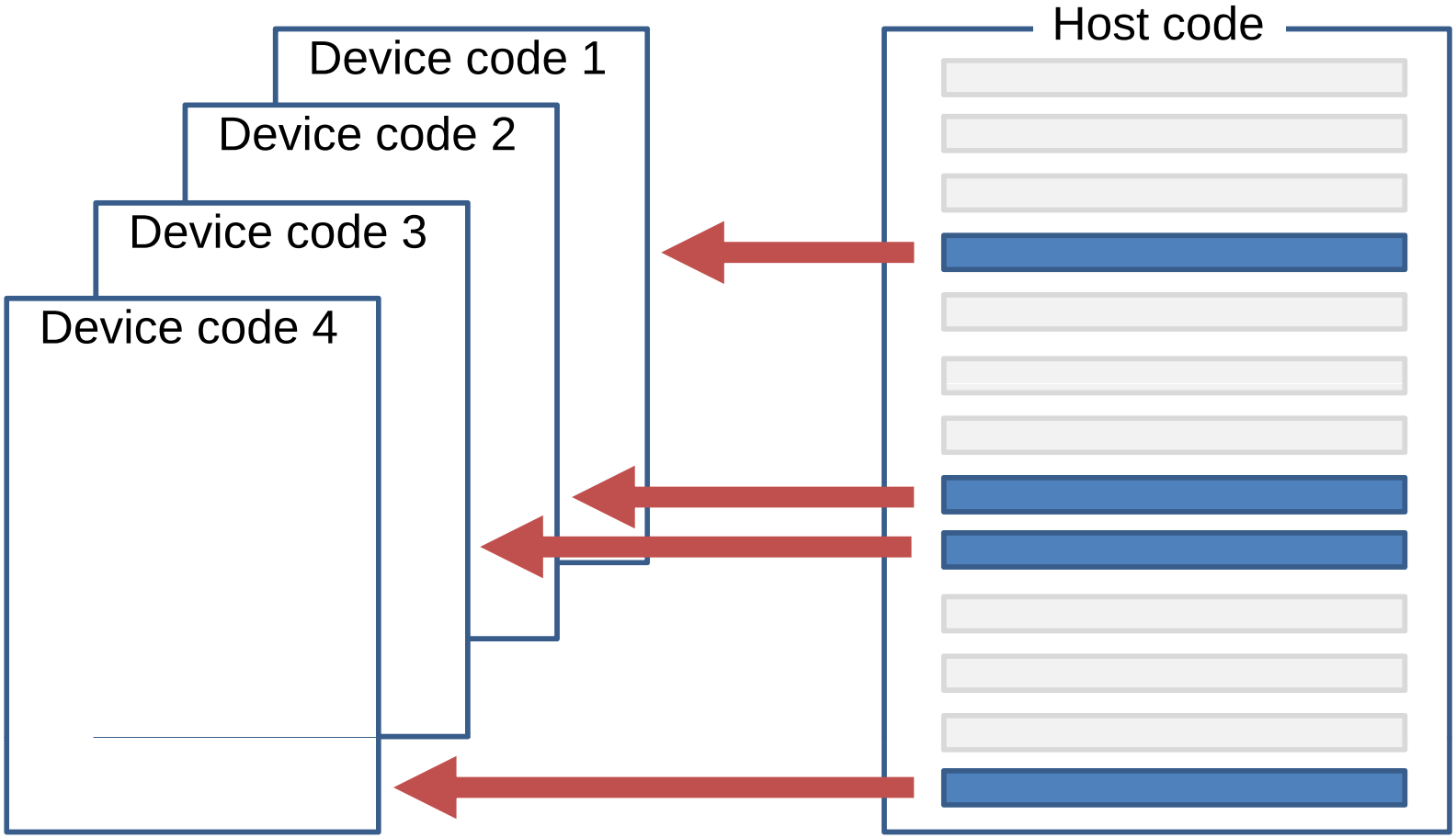
MIMD: Multiple Instruction Multiple Data Stream

（普通のマルチコア・プロセッサ
またはマルチ・プロセッサ。）

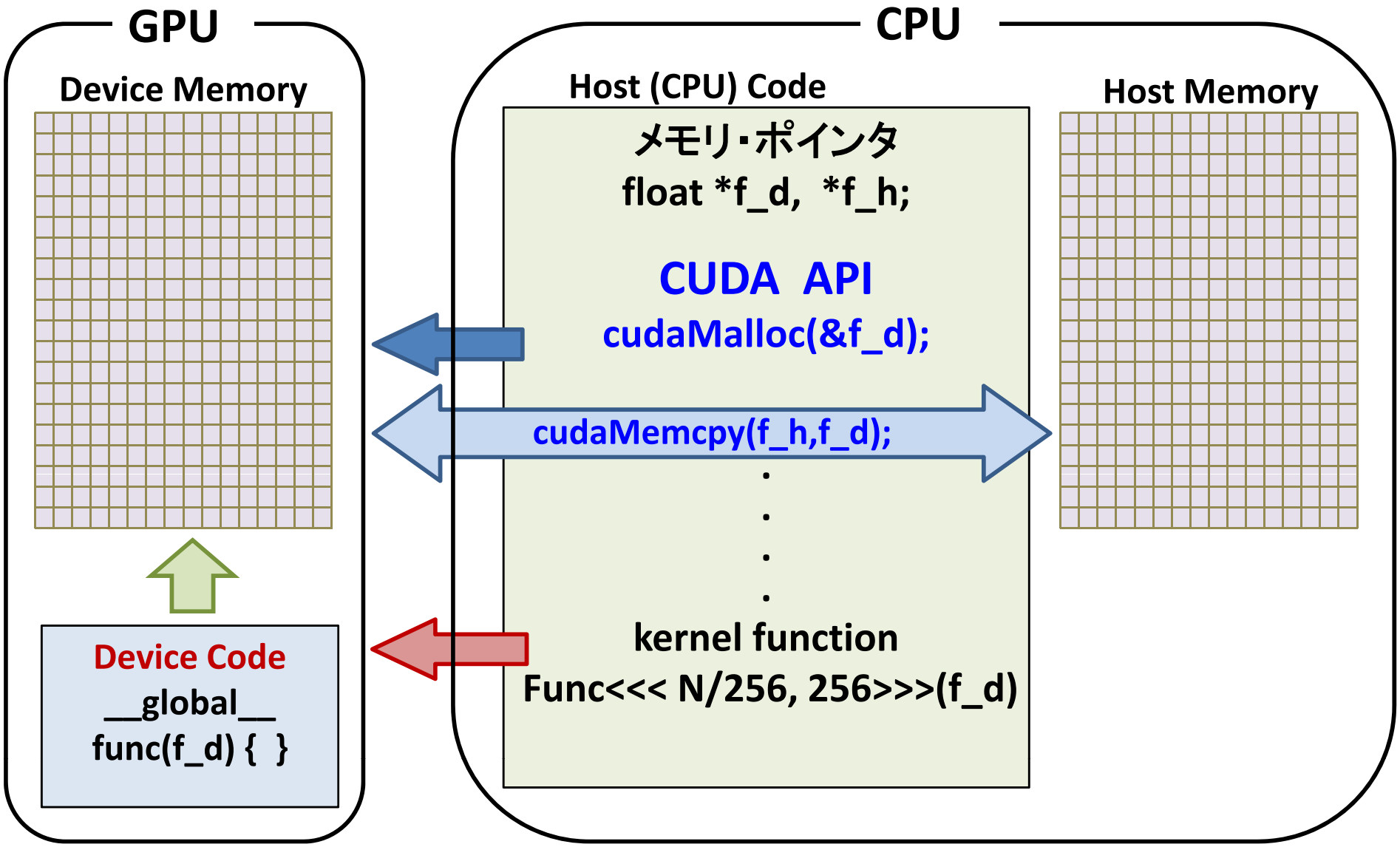
GPUコンピューティングの構造



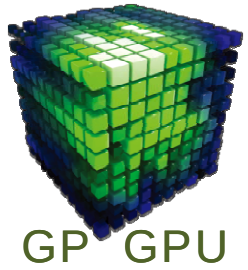
GPUは単独では動かない。Host からの指令が必要。
Host code と device code から構成される。



CUDAのプログラム実行の概念図



GPU kernel-function call



Host Code の中で call する。

kernel_function<<< Dg, Db, Ns, S>>>(a, b, c, ...);

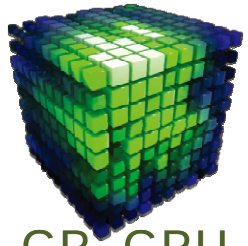
Dg: dim3 タイプの grid のサイズ指定
Db: dim3 タイプの block のサイズ指定
Ns: 実行時に指定する shared メモリのサイズ
省略可： 省略した場合は、0 が設定
S: 非同期実行の stream 番号
省略可： 省略した場合は、0 が設定され、
GPUのthread間は同期実行となる

Dg, Db で指定される数の Thread が実行される。

引数にHost memory のpointerの指定は不可。

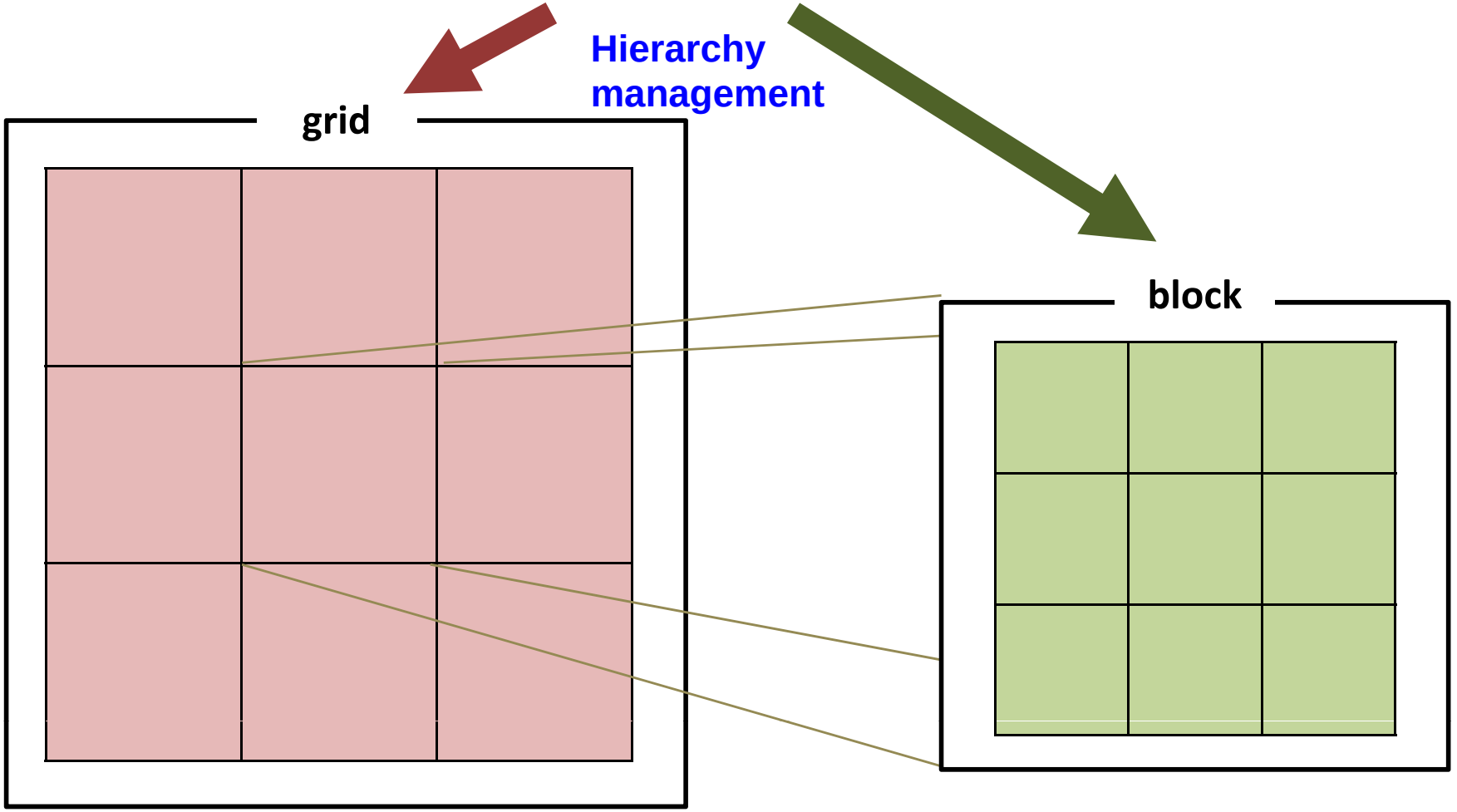
kernel function の実行は、CPU に対して絶えず非同期。

Thread Management

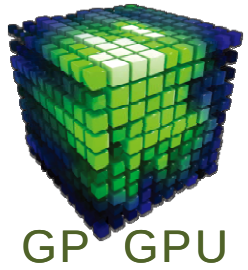


CPU: function(a, b, c, . . .);

GPU: function<<< grid, block >>>(a, b, c, . . .);



Example of CUDA programming



CPU Computing

```
for(j = 0; j < n; j++) A[j] = B[j];
```

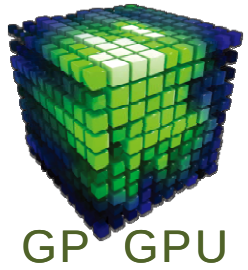
GPU Computing

```
__global__  void  copy(float *A, float *B)
{
    int j = blockDim.x*blockIdx.x + threadIdx.x;

    A[j] = B[j];
}
```

```
Launch GPU Kernel    copy <<< n/256,  256 >>> (A, B);
```

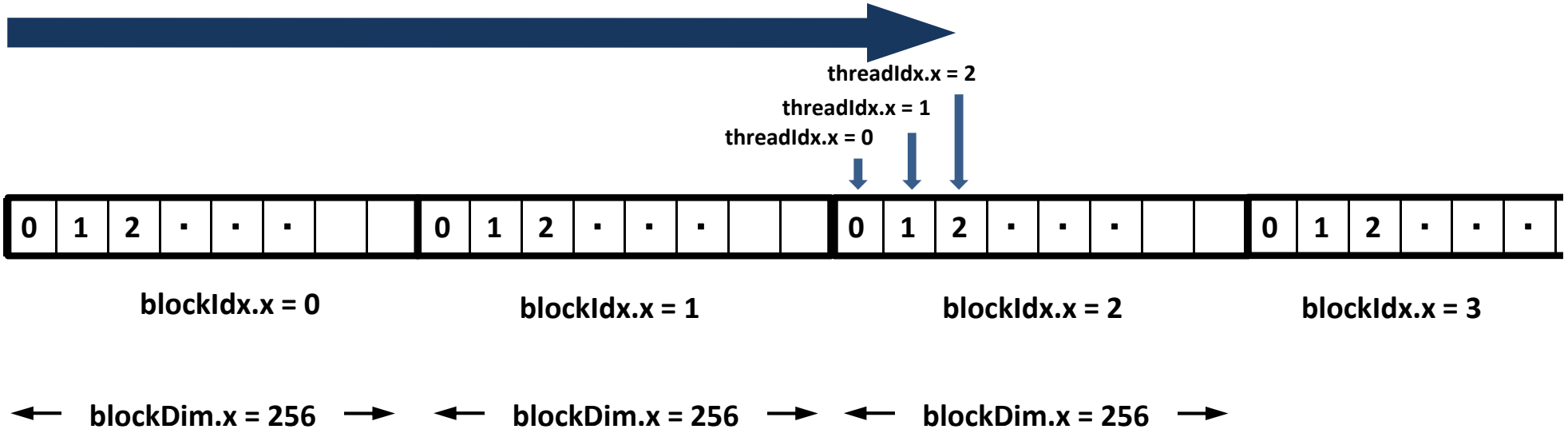
Parallel Data Access



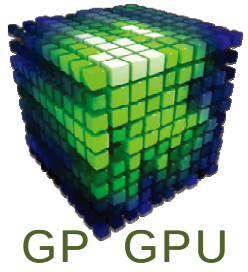
Access to 1-dimensional Array

1 element of the array from 1 thread

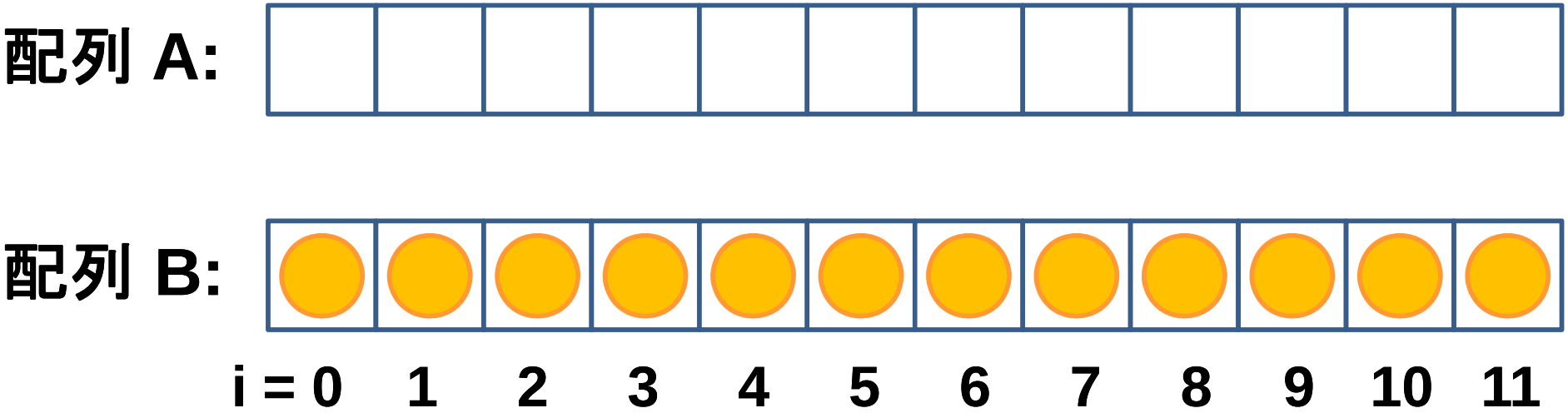
j = blockDim.x*blockIdx.x + threadIdx.x



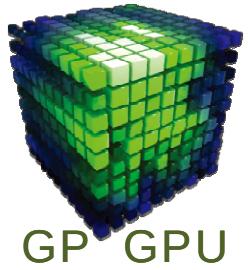
配列のコピー



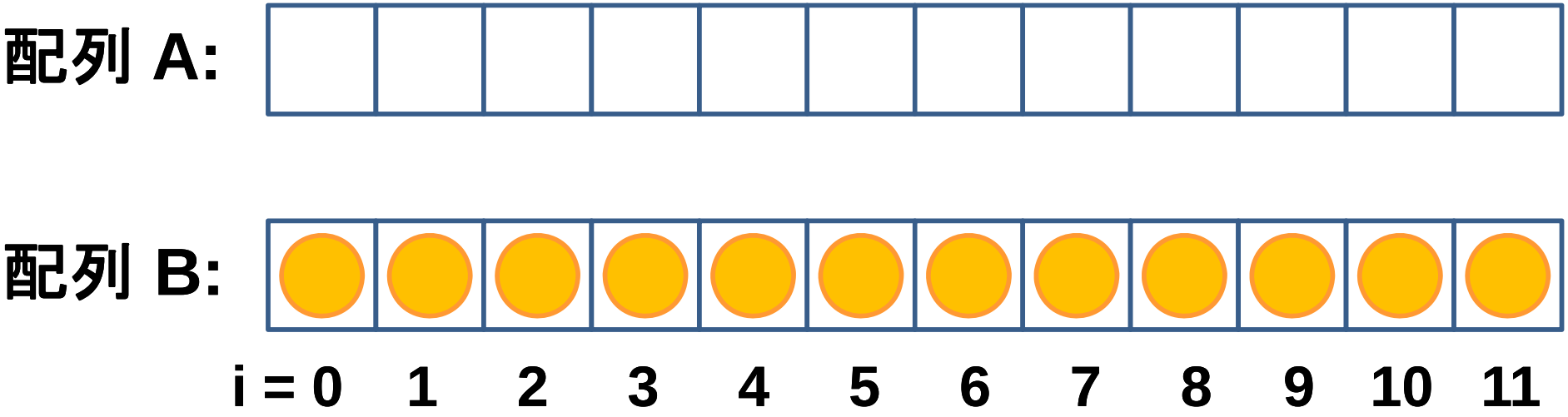
SISD: `for(i = 0; i < N; i++) A[i] = B[i];`



配列のコピー

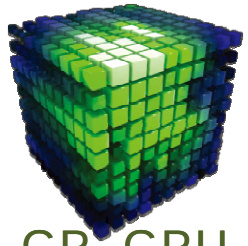


SIMD: 4個単位のスレッドが同じ命令(コピー)を実行する。



GPU は SIMD 命令を効率的に実行できるように最適化されたプロセッサ

まとめ≒アナロジー



Host code: 先生の立場になって、問題を多くの生徒にならせる指令を出す。
班分けを決める。

Device code: 個々の生徒の立場になって何を実行するかのプログラム。

クラスに何班あるか、自分は何班に所属しているか、班にはメンバーが何人いるか、自分はその班の何番目か。