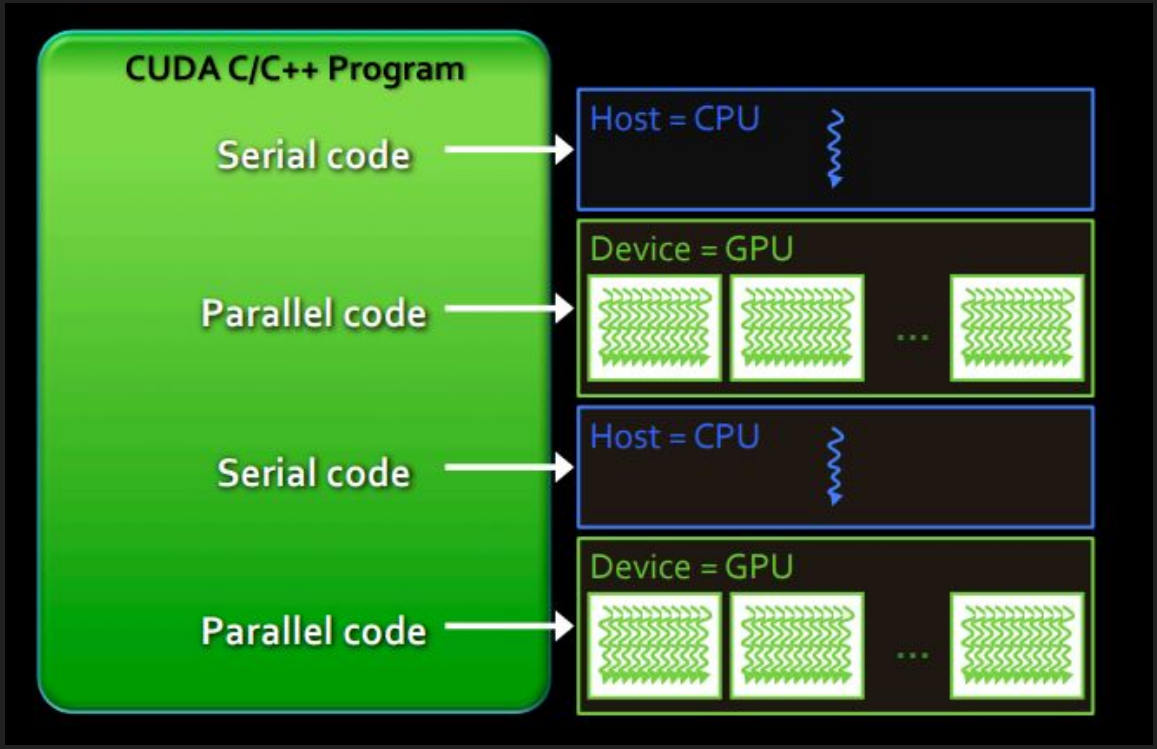
Introduction to CUDA

CUDA

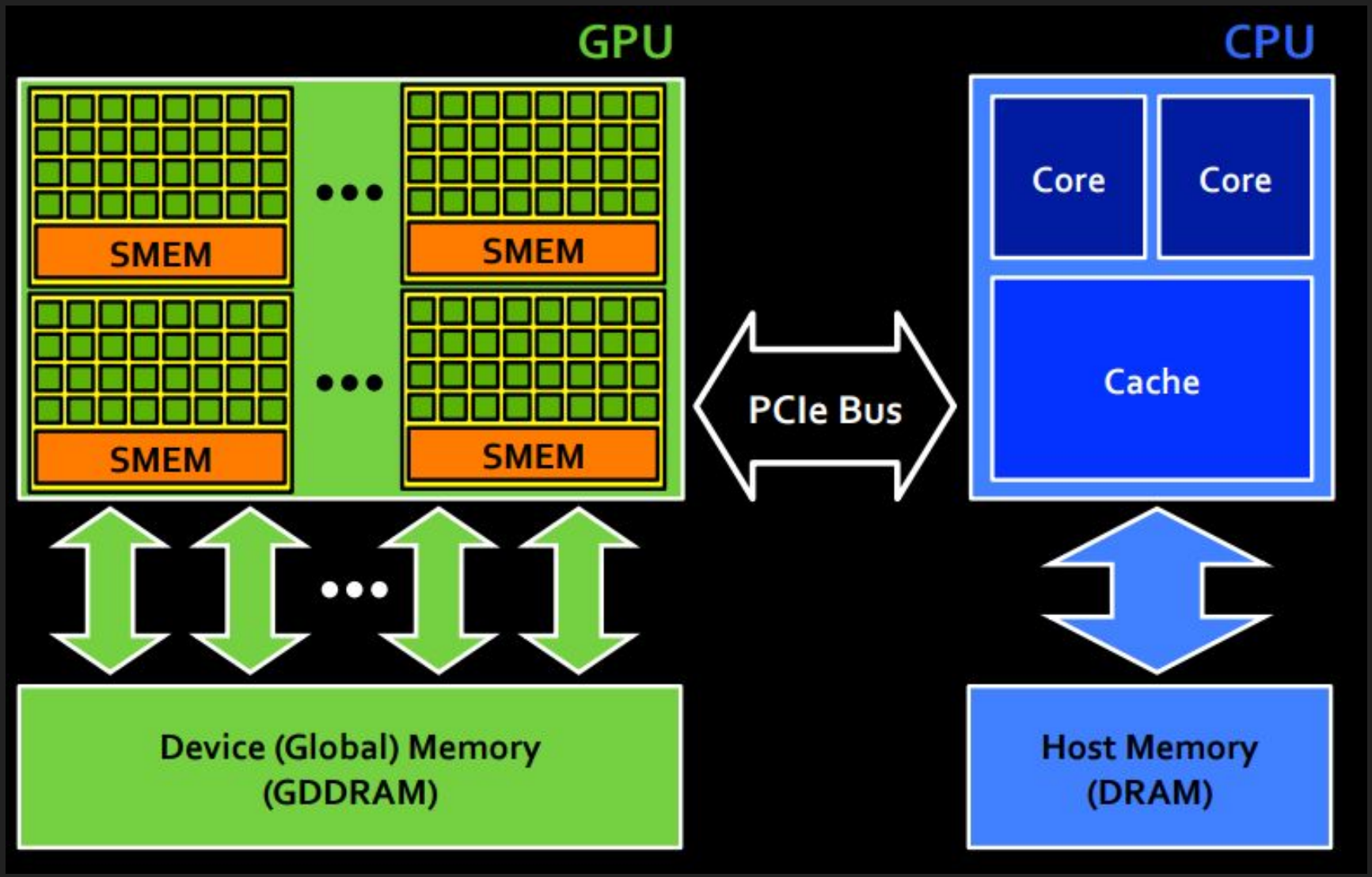
- Programing system for machines with GPUs
  - Programming Language
  - Compilers
  - Runtime Environments
  - Drivers
  - Hardware

Behavior of CUDA program

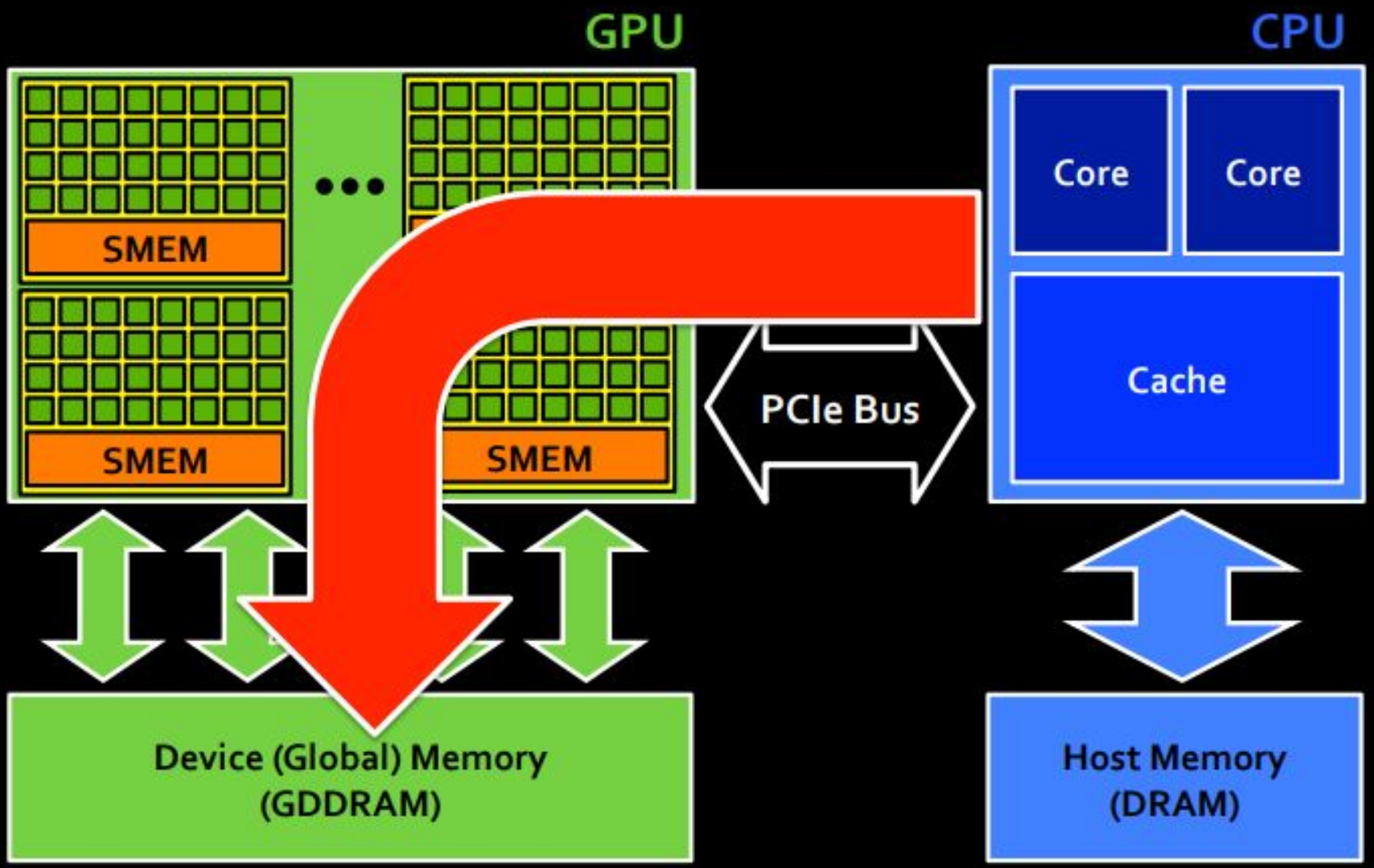
- Serial code executes in Host (CPU) thread
- Parallel code executes in many concurrent Device (GPU) threads across multiple parallel processing elements



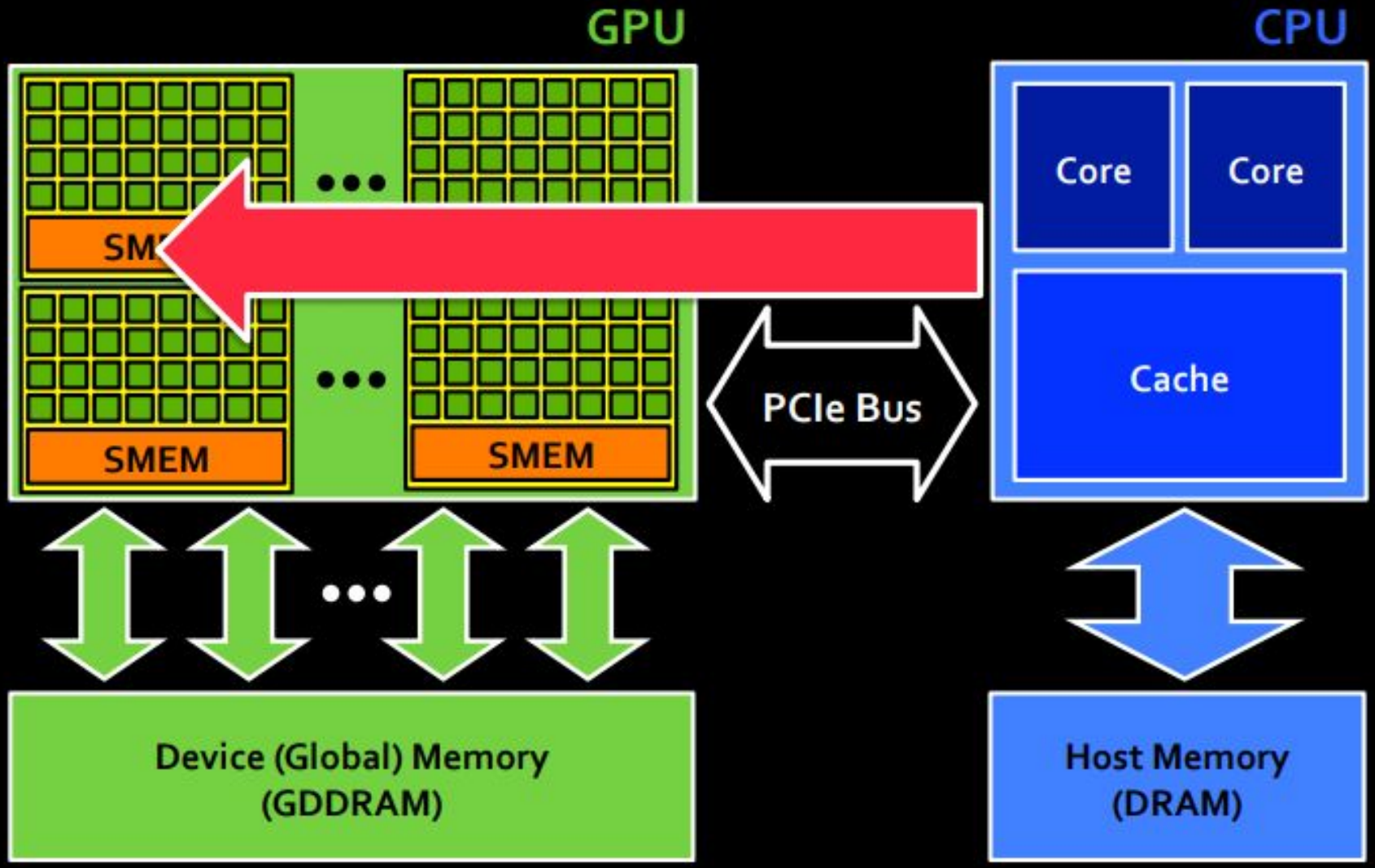
Execution flow



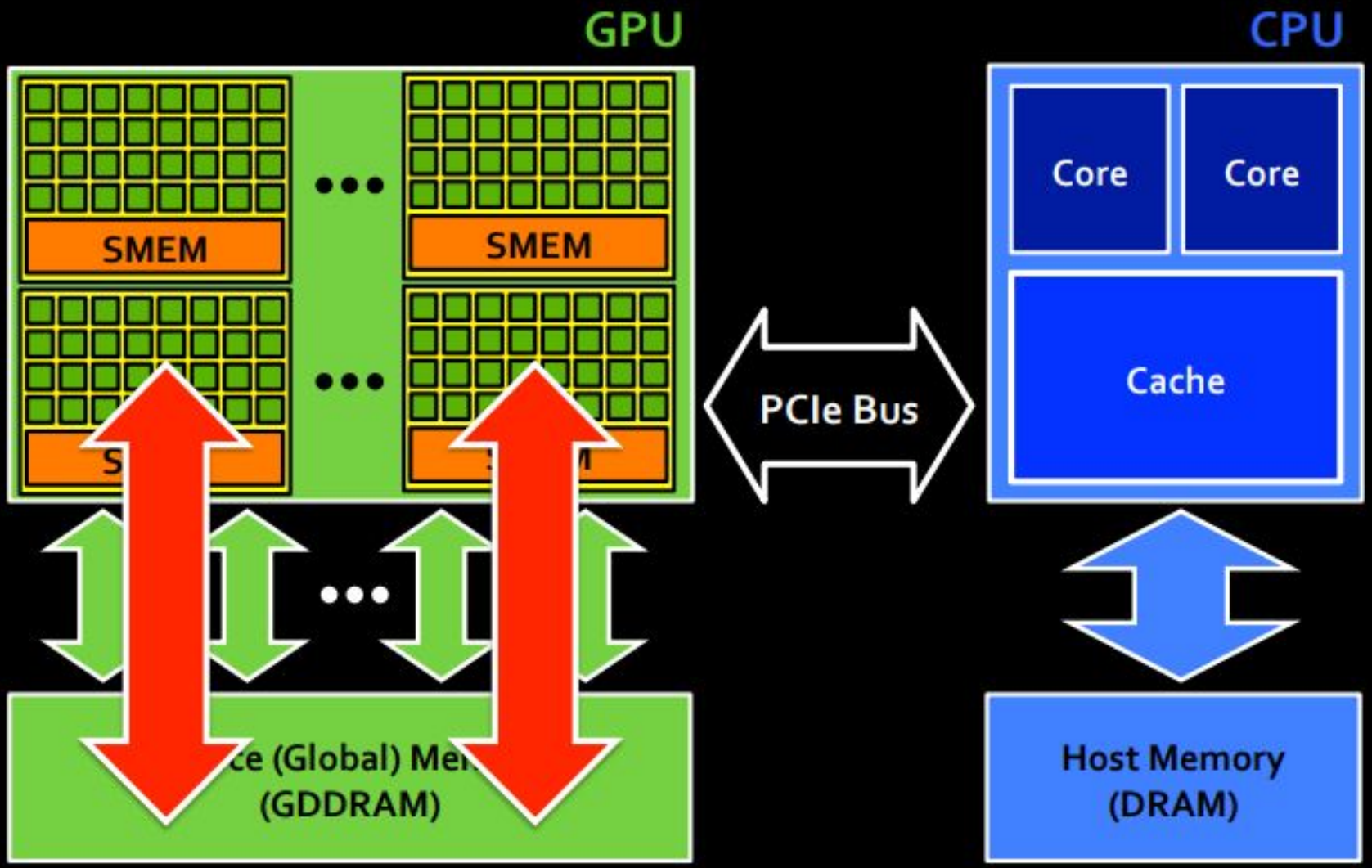
Step 1 – copy data to GPU memory



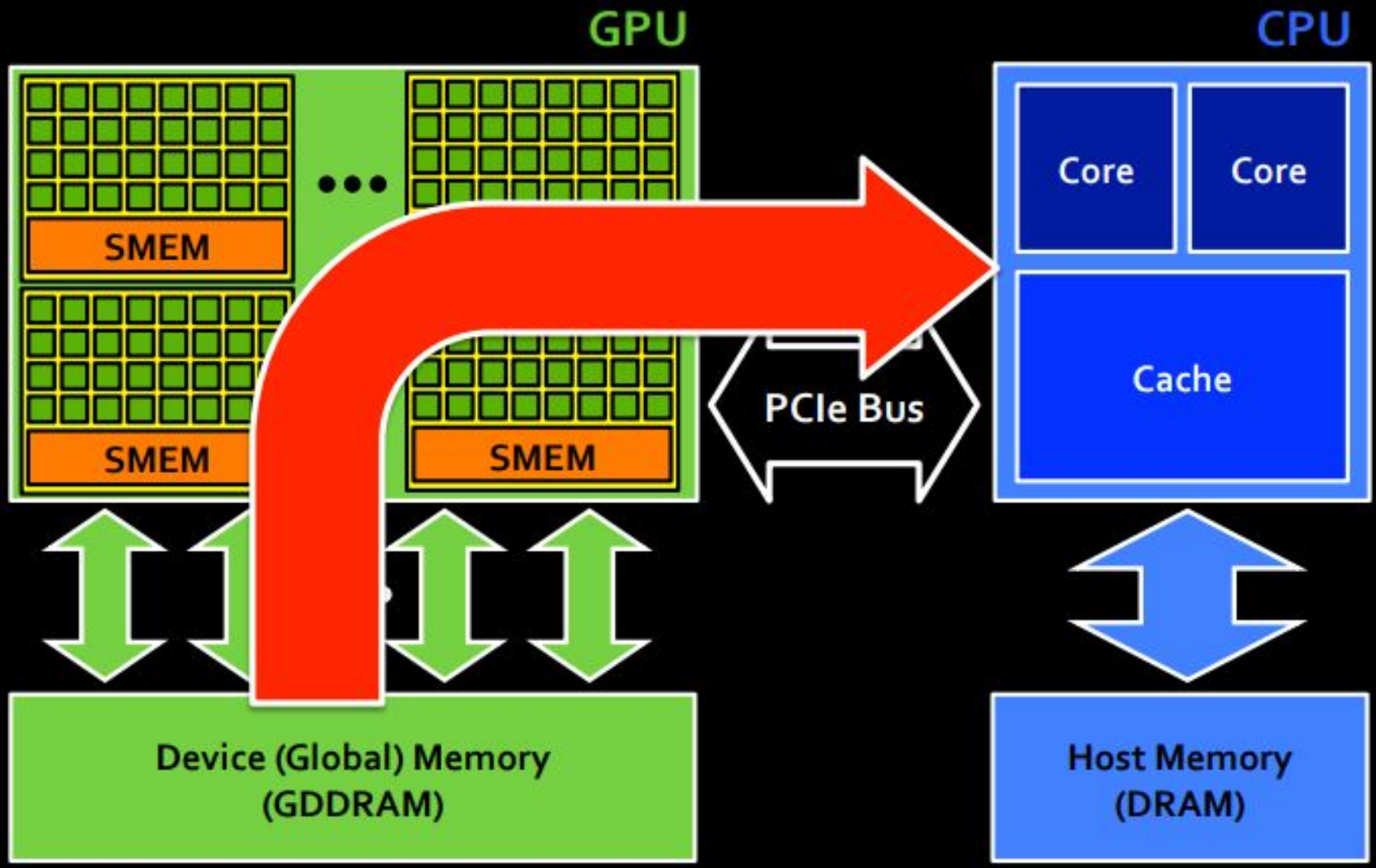
Step 2 – launch kernel on GPU



Step 3 – execute kernel on GPU



Step 4 – copy data to CPU memory

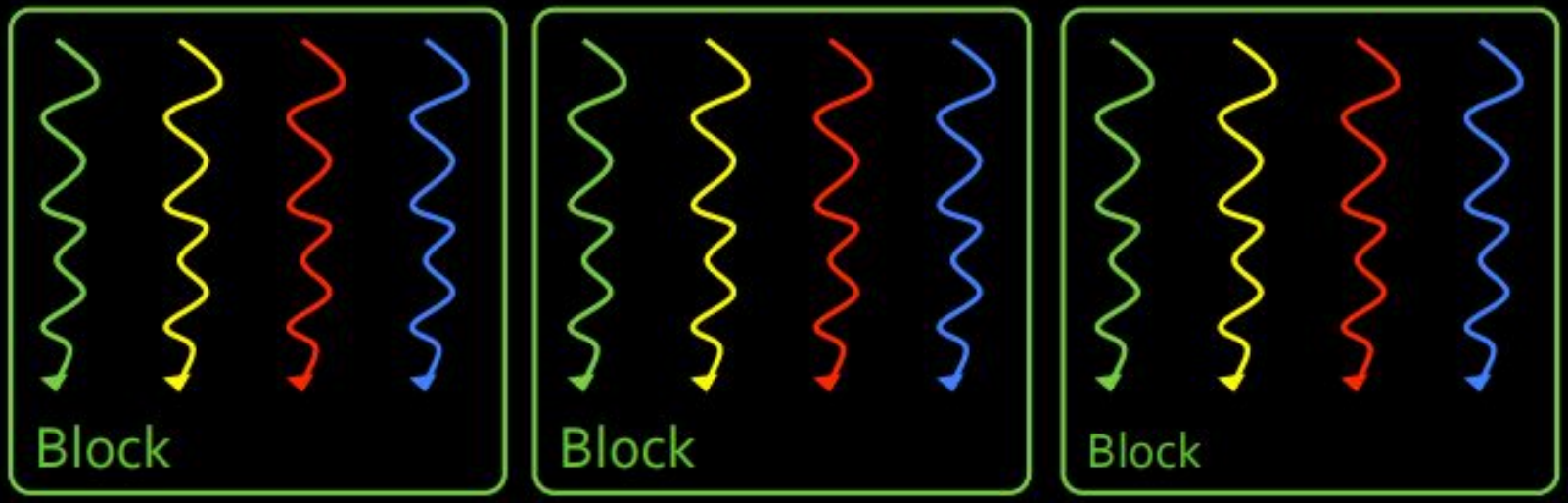


CUDA ARCHITECTURE

CUDA Thread Organization

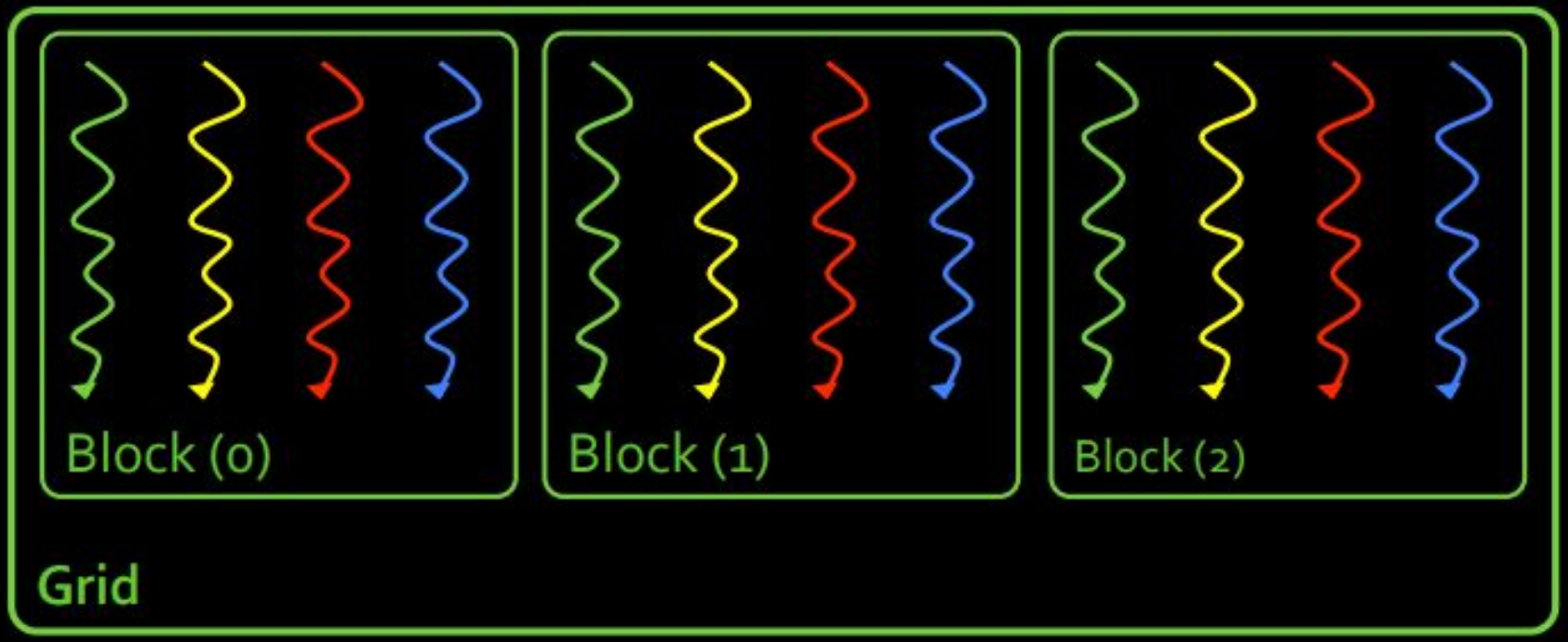
- GPUs can handle thousands of concurrent threads
- CUDA programming model supports even more
  - Allows a kernel launch to specify more threads than the GPU can execute concurrently
  - Helps to amortize kernel launch times

Blocks of threads



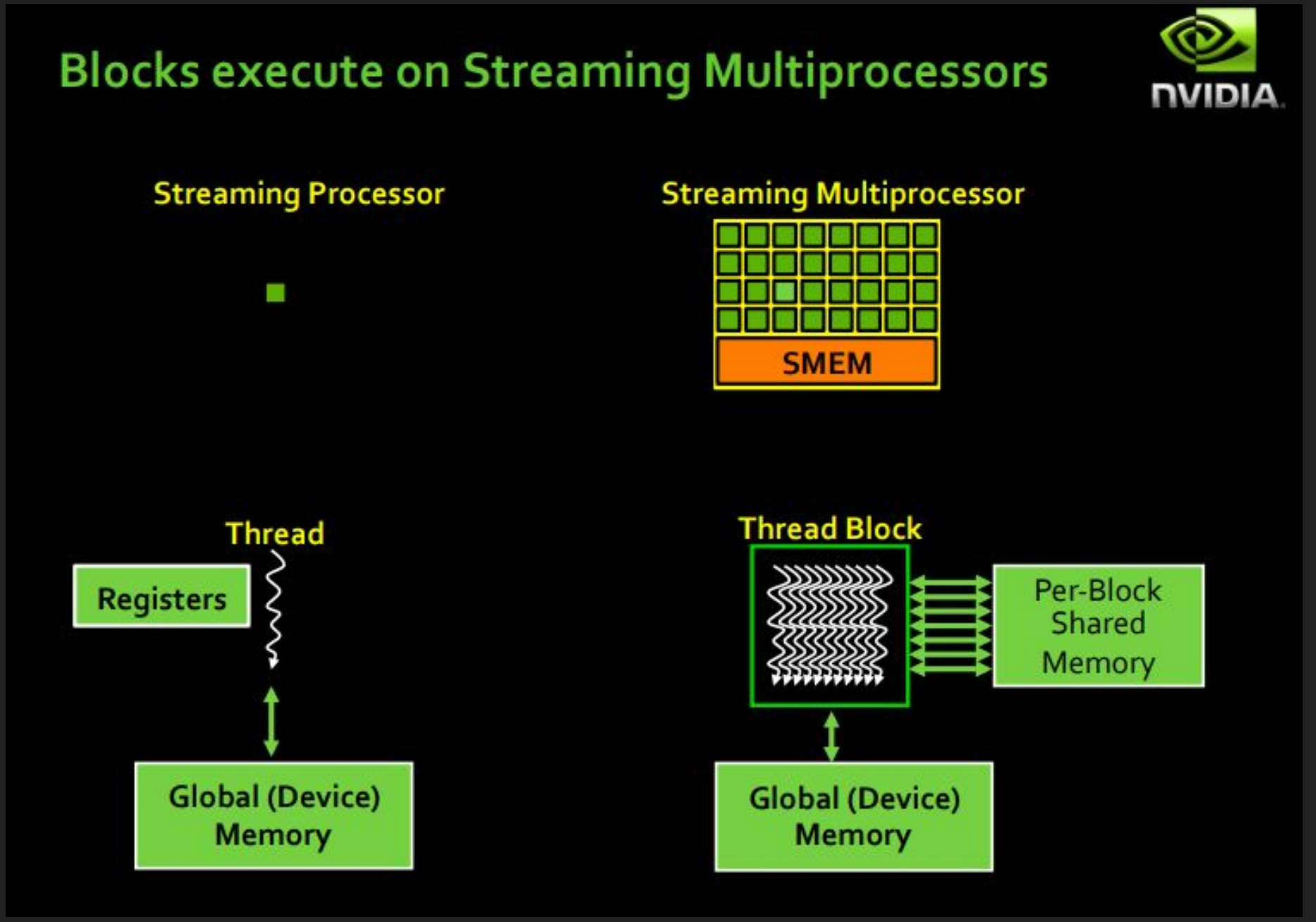
- Threads are grouped into blocks

Grids of blocks

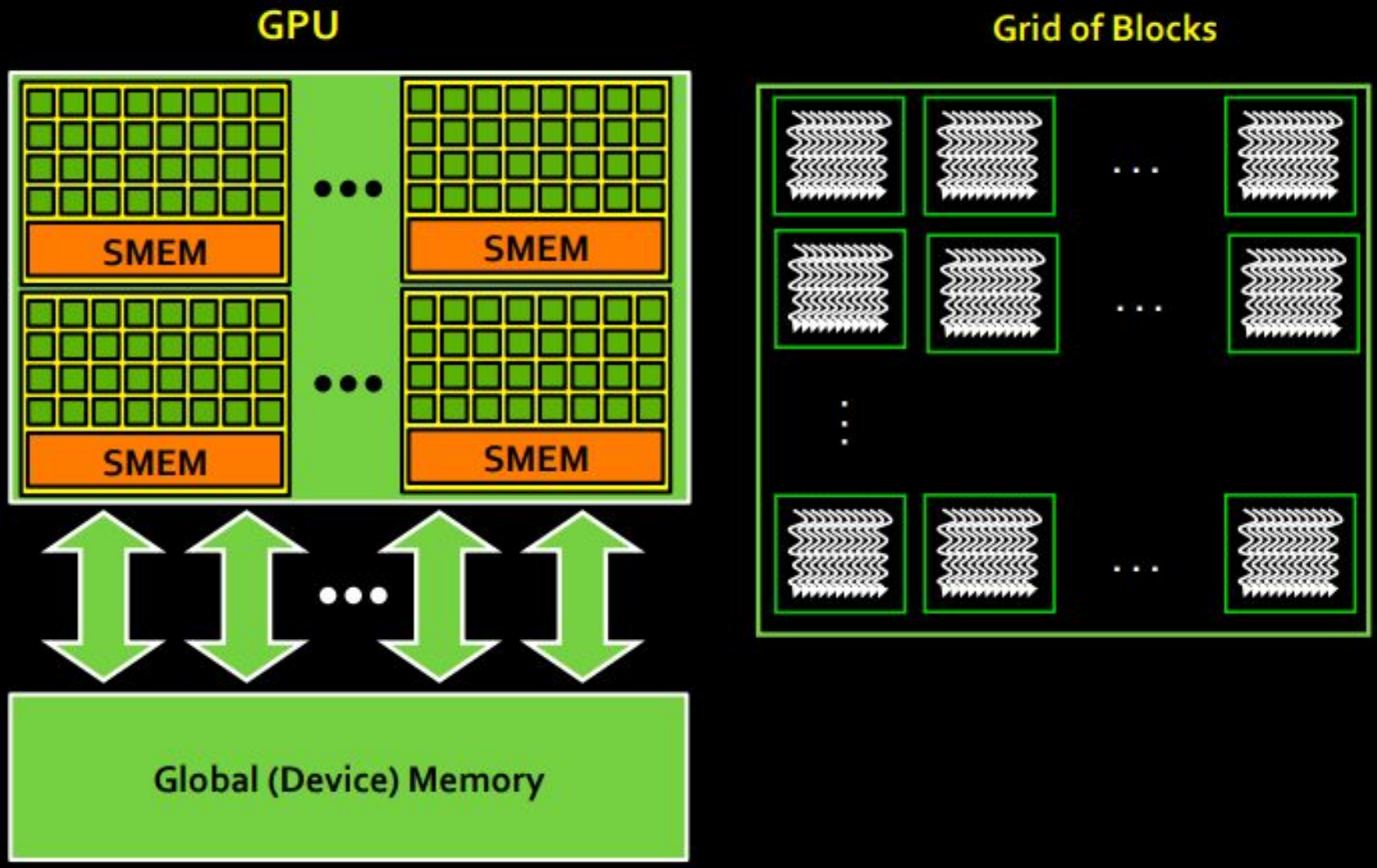


- Threads are grouped into blocks
- Blocks are grouped into a grid
- A kernel is executed as a grid of blocks of threads

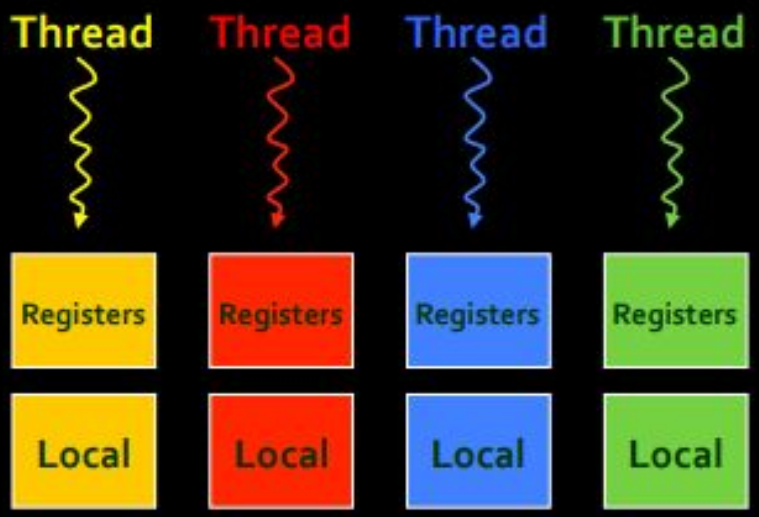
Blocks execute on Streaming Multiprocessors



Grids of blocks executes across GPU



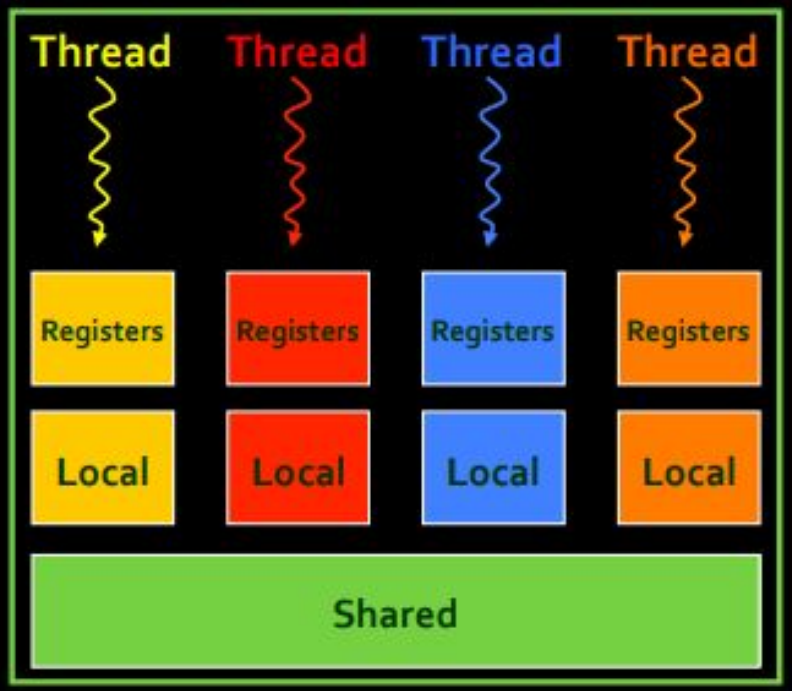
CUDA Memory Hierarchy



- Thread
  - Registers
  - Local memory

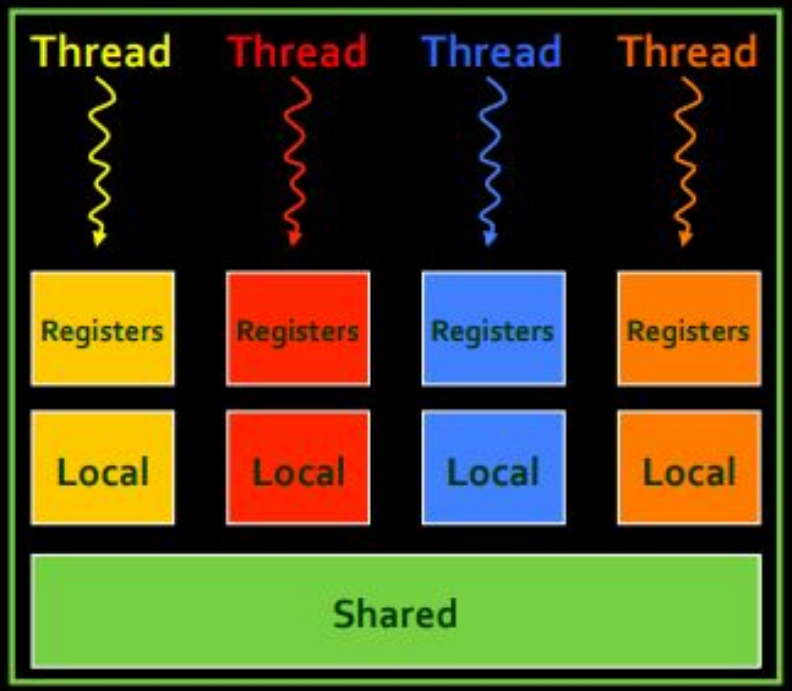
CUDA Memory Hierarchy

- Thread
  - Registers
  - Local memory
- Thread Block
  - Shared memory



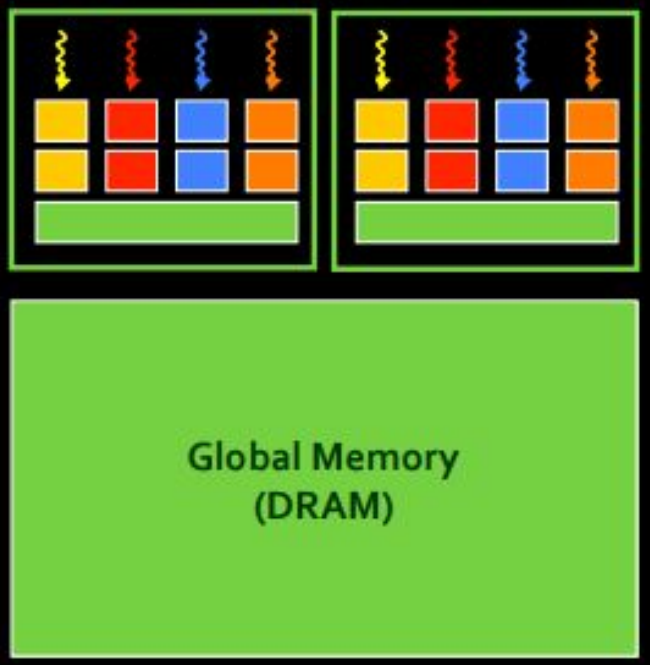
CUDA Memory Hierarchy

- Thread
  - Registers
  - Local memory
- Thread Block
  - Shared memory



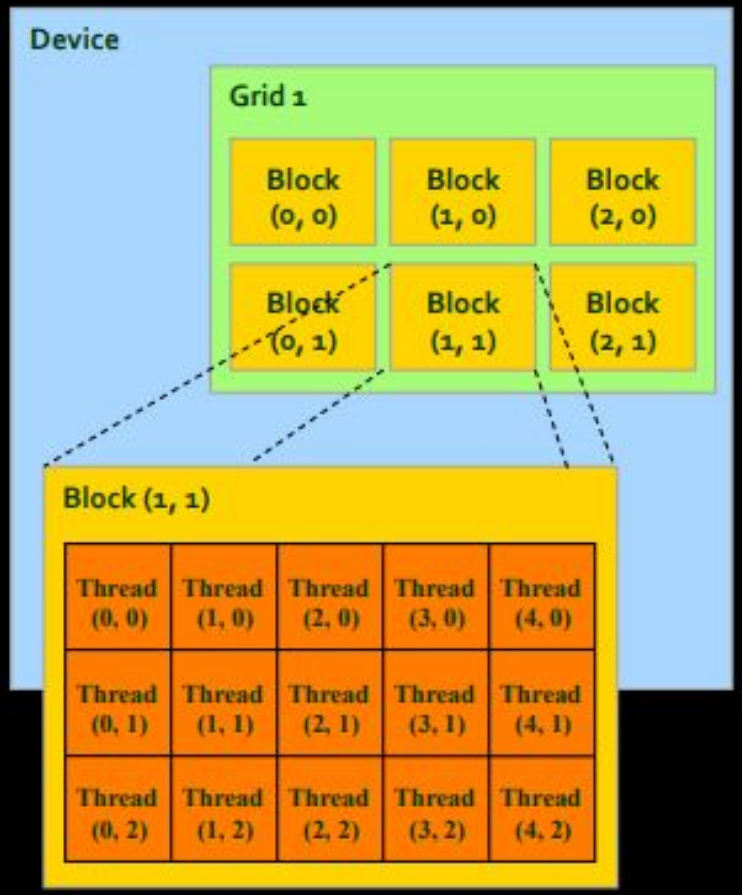
CUDA Memory Hierarchy

- Thread
  - Registers
  - Local memory
- Thread Block
  - Shared memory
- All Thread Blocks
  - Global Memory



Thread and Block ID and Dimensions

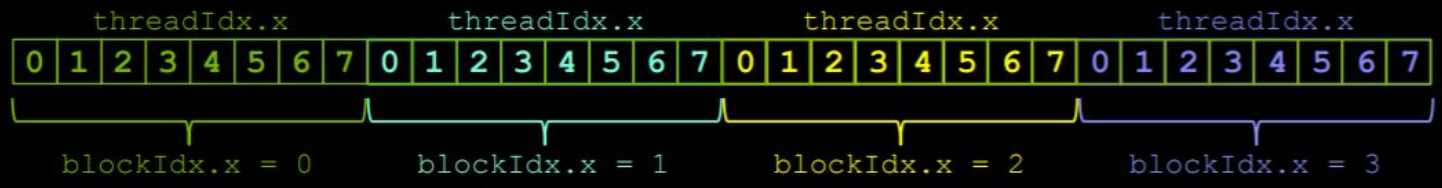
- Threads
  - 3D IDs, unique within a block
- Thread Blocks
  - 2D IDs, unique within a grid
- Dimensions set at launch
  - Can be unique for each grid
- Built-in variables
  - `threadIdx`, `blockIdx`
  - `blockDim`, `gridDim`
- Programmers usually select dimensions that simplify the mapping of the application data to CUDA threads



Indexing Arrays With Threads And Blocks

- No longer as simple as just using `threadIdx.x` or `blockIdx.x` as indices

- To index array with 1 thread per entry (using 8 threads/block)



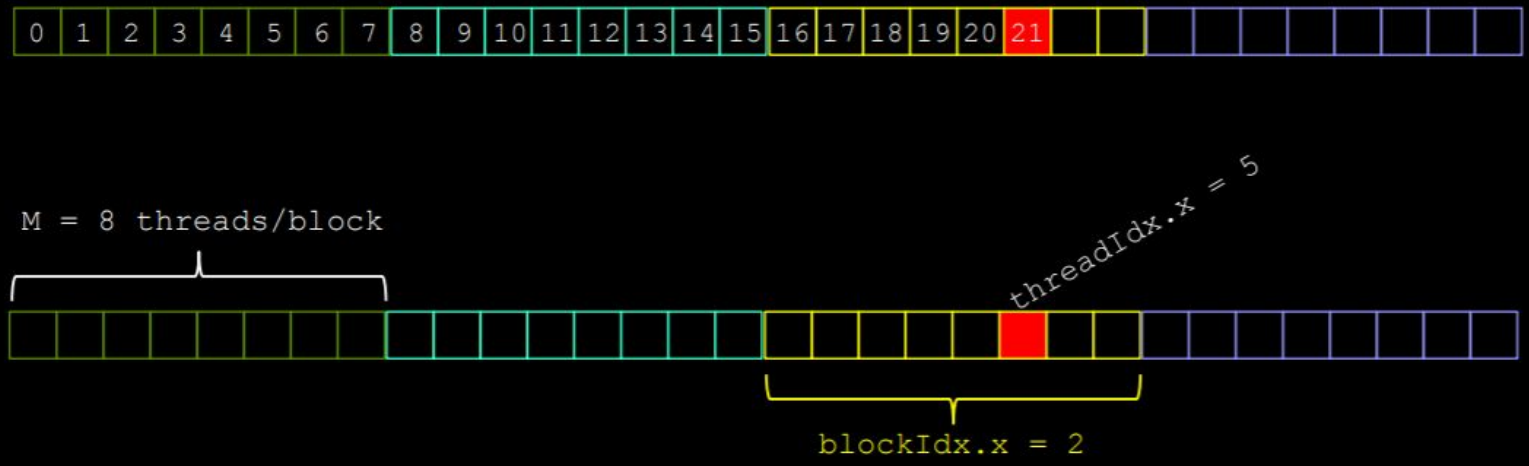
- If we have `M` threads/block, a unique array index for each entry given by

```
int index = threadIdx.x + blockIdx.x * M;

int index =       x     +     y      * width;
```

Indexing Arrays: Example

- In this example, the red entry would have an index of 21:




```
int index = threadIdx.x + blockIdx.x * M;
          =      5      +      2      * 8;
          = 21;
```