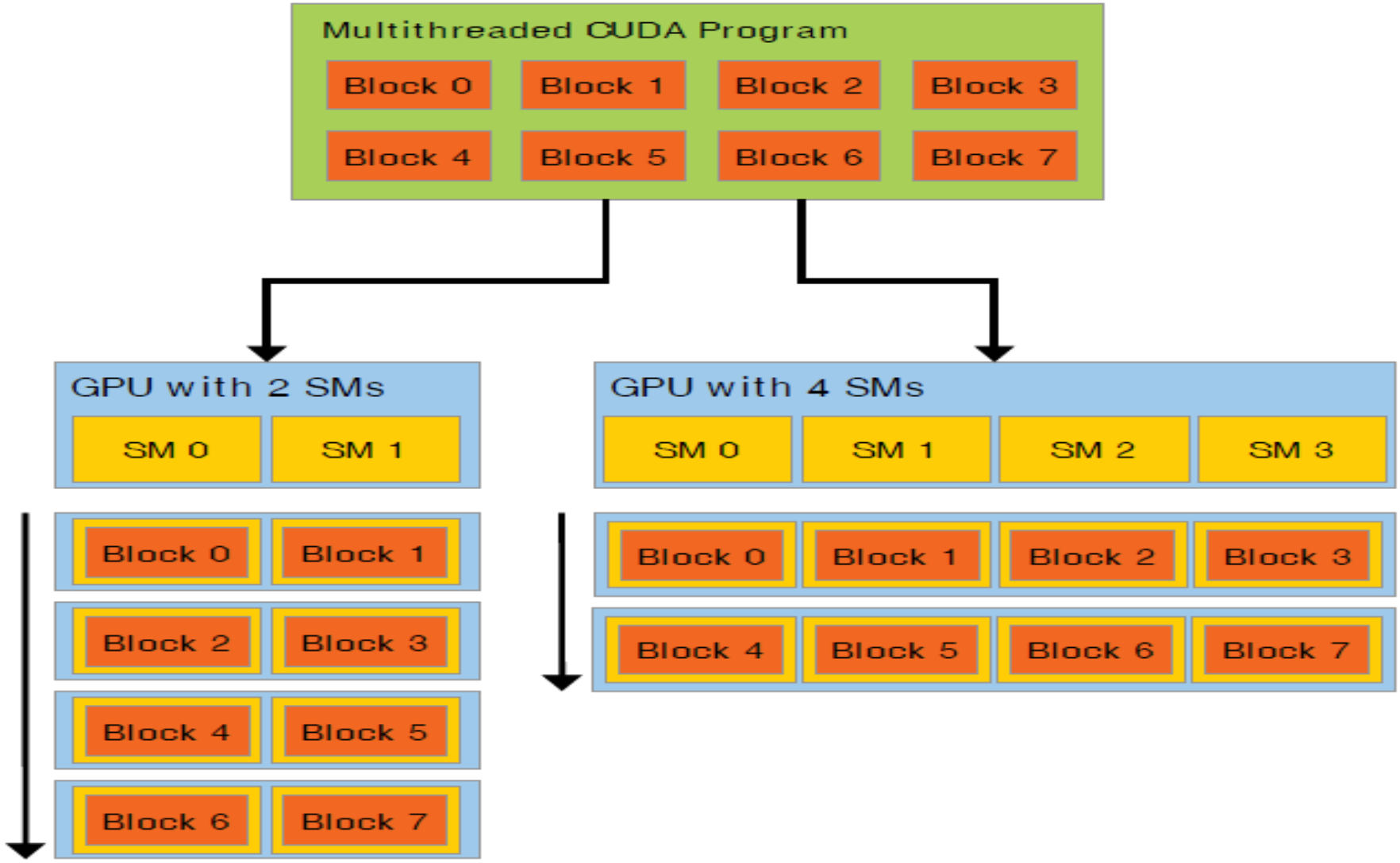
Lecture No.1: CUDA Thread Execution

Muhammad Osama Mahmoud, TA

Kernel Grid

- The kernel grid represents the workload of the device
- Consists of multiple blocks, each block can execute in any order relative to others
- The device hardware can assign blocks to any multiprocessors at any time to achieve best resource utilization possible
- Each multiprocessor can executes hundred of threads concurrently in block granularity
  - Up to 32 thread blocks per SM in Maxwell architecture
  - Up to 2048 threads per SM and 1024 threads per block

Kernel Grid Automatic Scalability



Thread Scheduling

- Kernel threads are grouped into small units executed on SMs called warps
- Each warp holds 32 threads
- Maximum number of warps assigned to single multiprocessor in Maxwell is increased to 64
- Warps are the scheduling units in SM
- Threads in a warp execute in SIMT model

Thread Scheduling (Cont.)

- Warps whose next instruction has its operands ready are eligible for execution
- Eligible warps are selected for execution on a prioritized scheduling policy
- All threads in a warp execute the instruction when their data are ready

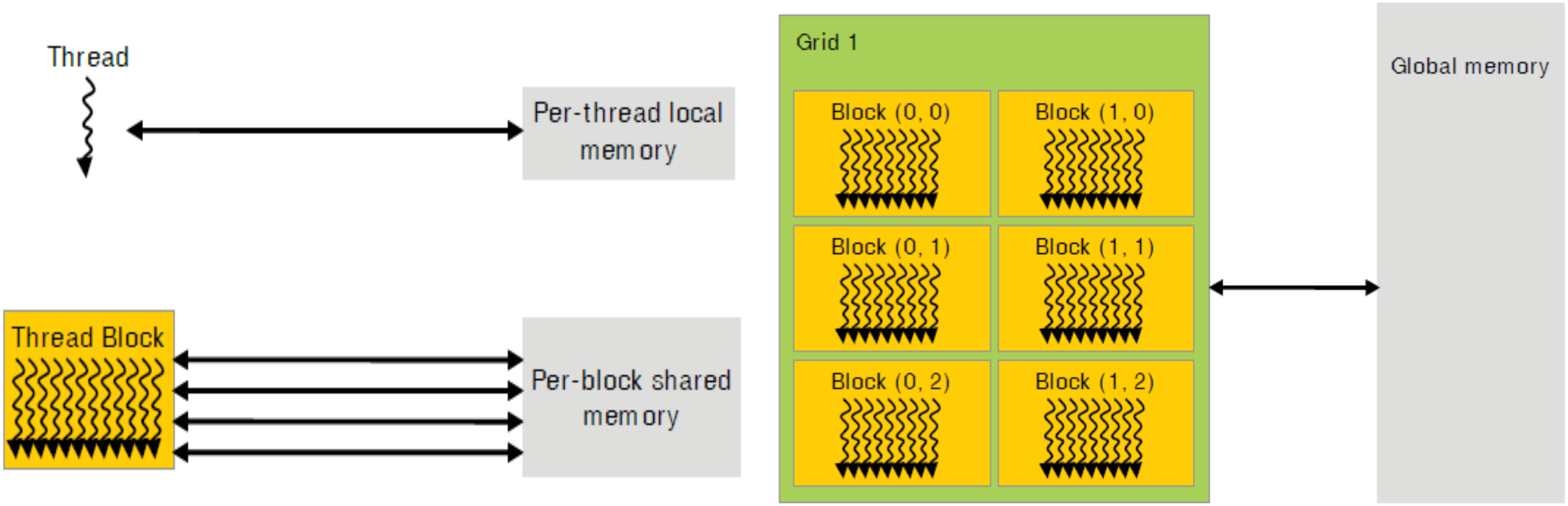
Control Divergence

- Arises in conditional branches
- A divergence occur when threads in s single warp take two different paths
- Different execution paths are serialized in current GPUs (no branch prediction in GPU control unit architecture)
- All execution paths traversed until no path remains
- A bottleneck in parallel execution performance

Control Divergence (Cont.)

- Example with divergence:
  - if (threadIdx.x < 15) {}, Thread indices less than 15 will follow a different path than the rest of threads in the same warp
- Example without divergence:
  - if (blockIdx.x < 15) {}, All threads reside in a warp will follow the same execution path for all thread blocks less than 15
- A general rule to avoid divergence as possible
  - Make the branch granularity a multiple of thread warps

CUDA Memory Hierarchy



Defining Memory Types

Variable Declaration		Memory	Scope	Lifetime
	int localVar;	Register	Thread	Thread
__shared__	int sharedVar;	Shared	Block	Block
__device__	int globalVar;	Global	Grid	Application
__constant__	int constVar;	Constant	Grid	Application

- Shared memory can hold up to 48 KB in Kepler and 64 in Maxwell architectures
- Constant memory are 64 KB ins size
- Automatic variables reside in registers by default except the arrays which reallocated in thread local memory (512 KB maximum)

Shared Memory

- Shared memory are visible only per threads block
- Threads must be synchronized after a write on shared memory
- Very fast compared to global memory
- Accessed by memory access instructions

Tiling Strategy

- Partition data into blocks or tiles fit into shared memory and use one thread block to handle each
- tile by:
  - Loading the tile from global memory to shared memory, using multiple threads
  - Perform the computation on the tile from shared memory, reducing traffic to the global memory
  - After completion, write results back from shared memory to global memory

References

[1] Wen-mei W. Hwu, “Heterogeneous Parallel Programming”. Online course, 2014. Available: https://class.coursera.org/hetero-002

[2] NVIDIA, “CUDA C Programming Guide”, June 2014.