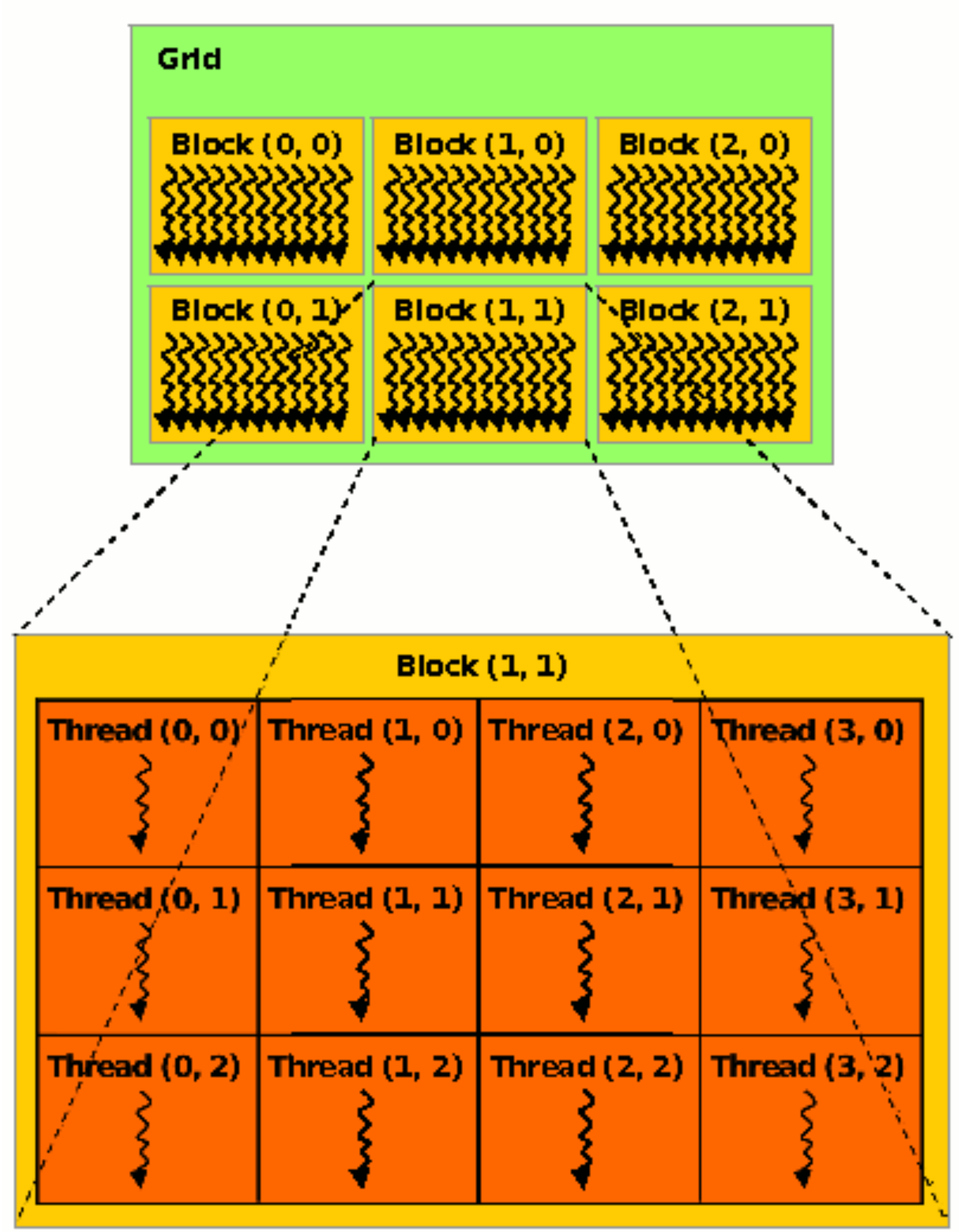
NVIDIA GPU Memory Hierarchy

- Grids map to GPUs
- Blocks map to the MultiProcessors (MP)
- Threads map to Stream Processors (SP)
- Warps are groups of (32) threads that execute simultaneously



NVIDIA GPU Memory Architecture

- In a NVIDIA GTX 480:
  - Maximum number of threads per block: 1024
  - Maximum sizes of x-, y-, and z- dimensions of thread block: 1024 x 1024 x 64
  - Maximum size of each dimension of grid of thread blocks: 65535 x 65535 x 65535

Defining Grid/Block Structure

- Need to provide each kernel call with values for two key structures:
  - Number of blocks in each dimension
  - Threads per block in each dimension

- myKernel<<< B,T >>>(arg1, …);

- B – a structure that defines the number of blocks in grid in each dimension (1D or 2D).
- T – a structure that defines the number of threads in a block in each dimension (1D, 2D, or 3D).

1D Grids and/or 1D Blocks

- If want a 1-D structure, can use a integer for B and T in:
- myKernel<<< B, T >>>(arg1, …);
- B – An integer would define a 1D grid of that size
- T – An integer would define a 1D block of that size

- Example: myKernel<<< 1, 100 >>>(arg1, ...);

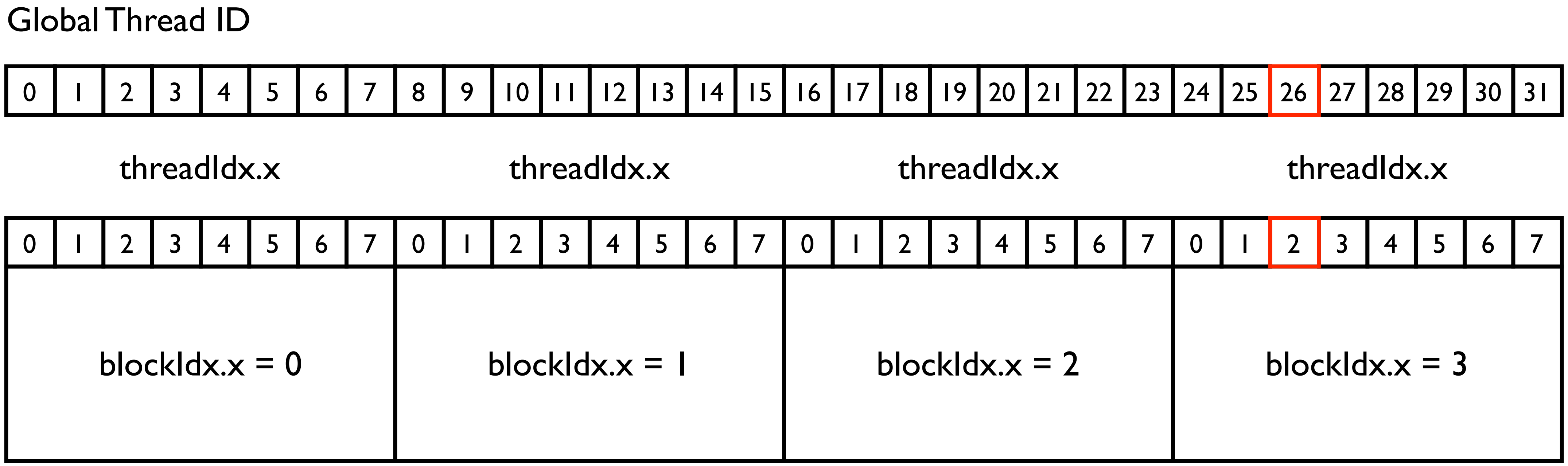
CUDA Built-In Variables

- **blockIdx.x**, **blockIdx.y**, **blockIdx.z** are built-in variables that returns the block ID in the x-axis, y-axis, and z-axis of the block that is executing the given block of code.
- **threadIdx.x**, **threadIdx.y**, **threadIdx.z** are built-in variables that return the thread ID in the x-axis, y-axis, and z-axis of the thread that is being executed by this stream processor in this particular block.
- **blockDim.x**, **blockDim.y**, **blockDim.z** are built-in variables that return the "block dimension" (i.e., the number of threads in a block in the x-axis, y-axis, and z-axis).

- So, you can express your collection of blocks, and your collection of threads within a block, as a 1D array, a 2D array or a 3D array.
- These can be helpful when thinking of your data as 2D or 3D.

- The full global thread ID in x dimension can be computed by:
  - x = blockIdx.x * blockDim.x + threadIdx.x;

Thread Identification Example: x-direction



Global Thread ID

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

threadIdx.x								threadIdx.x								threadIdx.x								threadIdx.x							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
blockIdx.x = 0								blockIdx.x = 1								blockIdx.x = 2								blockIdx.x = 3							

- Assume a hypothetical 1D grid and 1D block architecture: 4 blocks, each with 8 threads.

- For Global Thread ID 26:
  - gridDim.x = 4 x 1
  - blockDim.x = 8 x 1
  - Global Thread ID = blockIdx.x * blockDim.x + threadIdx.x
  - = 3 x 8 + 2 = 26