4400 review - CUDA

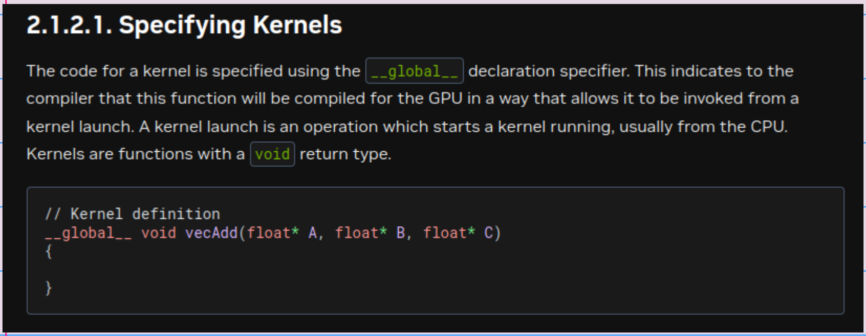
CUDA hierarchy:

1.) threads
2.) blocks
3.) grids

} Can be 1D, 2D, 3D

CPU : host GPU: device

Kernel is specified with a
__global__ specifier



2.1.2.1. Specifying Kernels

The code for a kernel is specified using the `__global__` declaration specifier. This indicates to the compiler that this function will be compiled for the GPU in a way that allows it to be invoked from a kernel launch. A kernel launch is an operation which starts a kernel running, usually from the CPU. Kernels are functions with a `void` return type.

```
// Kernel definition
__global__ void vecAdd(float* A, float* B, float* C)
{

}
```

vecAdd <<< numBlocks, numThreads >>> (A, B, C)

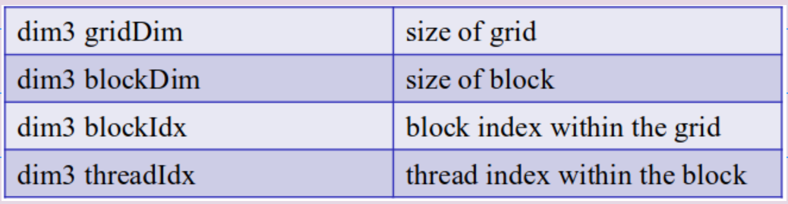
will launch vecAdd with that many blocks
and threads.

Global index of a thread:

1D: idx = blockDim.x * blockIdx.x + threadIdx.x
2D: idx_x = blockDim.x * blockIdx.x + threadIdx.x
idx_y = blockDim.y * blockIdx.y + threadIdx.y

Then use row major ordering to calculated a 1D
index on flattened 2D array.
width = blockDim.x * GridDim.x

dim3 is a CUDA datatype:



dim3 gridDim	size of grid
dim3 blockDim	size of block
dim3 blockIdx	block index within the grid
dim3 threadIdx	thread index within the block

cudaDeviceSynchronize() after each iteration

TECHNICALLY, this is inefficient. But this is the way that was taught to us.