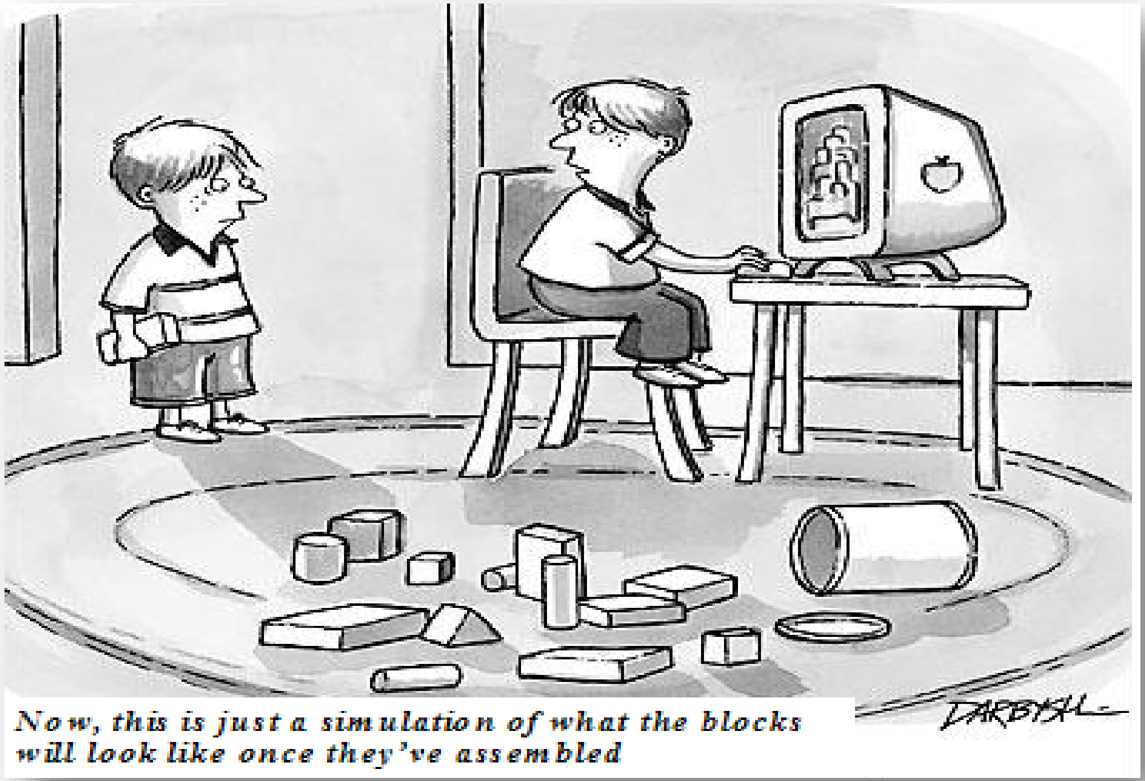
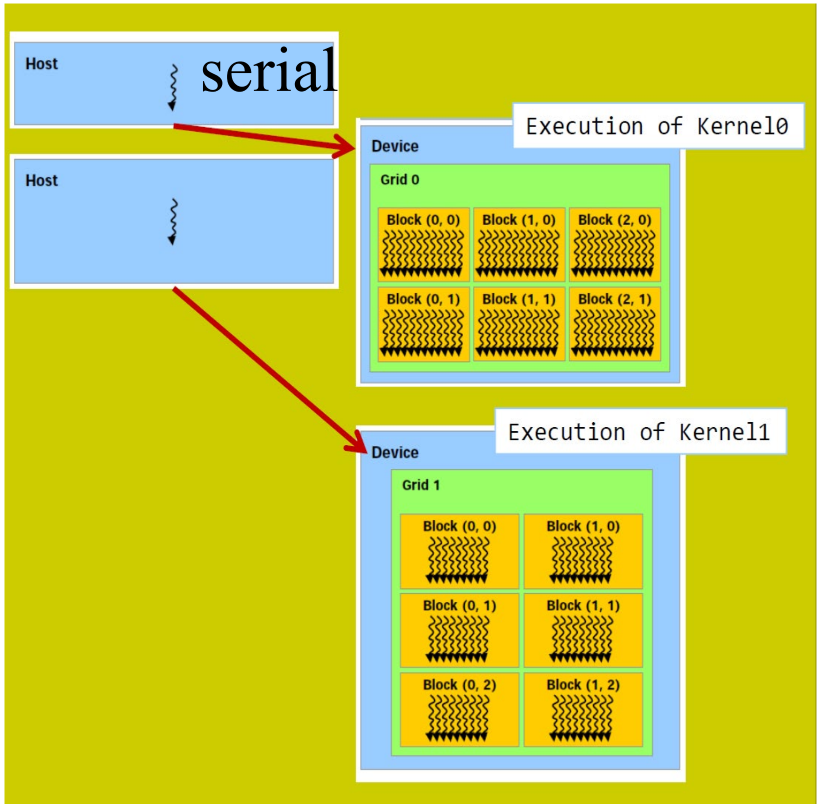
ECE569
Module 7


Now, this is just a simulation of what the blocks will look like once they've assembled
DARBYSHIRE

- Thread Organization (Grid, Block)

CUDA Execution Model is Asynchronous

- **Heterogeneous host (CPU) + device (GPU) application C program**
  - Serial parts in host C code
  - Parallel parts in device SPMD kernel code



By default, execution on host doesn't wait for kernel to finish

Quick check

- **What is GPU good at?**

❑ Launching a small number of threads efficiently

❑ Launching a large number of threads efficiently

❑ Running one thread very quickly

❑ Running one thread that does lots of work in parallel

❑ Running a large number of threads in parallel

Vector Addition Host Code: Kernel Launch

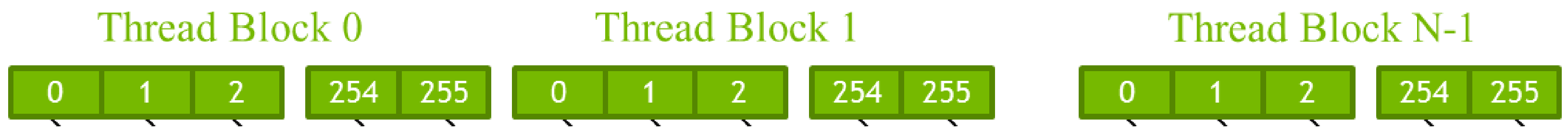
```
void vecAdd(float *h_A, float *h_B, float *h_C, int n){
   int size = n * sizeof(float);
   float *d_A, *d_B, *d_C;

   cudaMalloc((void **) &d_A, size);
   cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
   cudaMalloc((void **) &d_B, size);
   cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
   cudaMalloc((void **) &d_C, size);

   // Kernel invocation:
   // vecAddKernel <<<grid,block>>>(params)

   cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);

   cudaFree(d_A); cudaFree(d_B); cudaFree (d_C);
}
```

Launching the kernel from host

- kernel_routine<<<gridDim, blockDim>>>(args);
  - gridDim is the number of instances of the kernel
    - (the “grid” size = number of blocks)
  - blockDim is the number of threads within each instance
    - (the “block” size)
  - args is a limited number of arguments, usually mainly pointers to arrays in graphics memory, and some constants which get copied by value

Thread Blocks: Scalable Cooperation

- kernel_routine<<<10, 256>>>(args);
  - Number of blocks =10, Number of threads per block = 256



- **Thread position: Which block, which thread?**
  - Need a block identifier: blockIdx.**x**
  - Need a thread identifier: threadIdx.**x**
    - .x notation
      - In x direction (can be 3D: x,y,z)
  - Analogy
    - blockIdx.x: area code, threadIdx.x: local phone number
  - Mapping function
    - Id = blockIdx.x * blockDim.x + threadIdx.x

Example Array Indexing

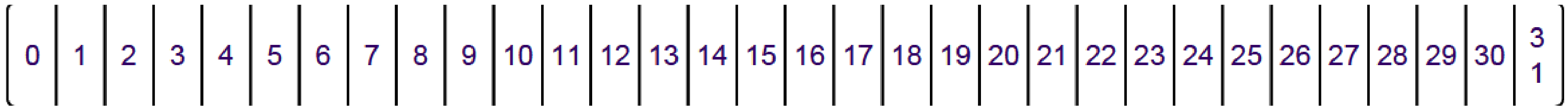
- Consider indexing into an array A, where one thread is accessing one element. Assume that you launch **8** threads per block and the array is 32 entries long. Which element of the A array will be processed by the thread **Id** of index 5 in block of index 2?

❑ 13

❑ 21

❑ 29

❑ 42



Grid Size and Block Size

- **1D grid with 4 blocks, each with 64 threads:**
  - gridDim = 4
  - blockDim = 64
  - blockIdx ranges from 0 to 3
  - threadIdx ranges from 0 to 63

Next

- **Writing kernel code**
  - Vector addition