**Introduction to GPUs cuda\_example\_code.cu**

**#include <stdlib.h>  
#include <cuda.h>**

**// This is a kernel - it executes on the device (GPU), once per thread. Programmer's   
// job is to write the kernel code from the standpoint of a single thread. The special  
// variable 'threadIdx.x' is automatically provided - it tells you what your thread  
// number is. In this example we call example\_scalar\_function on a single data array  
// element.**

**\_\_global\_\_ my\_kernel(float\* d\_data) {  
 int tid = threadIdx.x;  
 d\_data[tid] = example\_scalar\_function(d\_data[tid]);  
}**

**// This is calling code that executes on the host (CPU). The purpose of this example  
// function is to evaluate in parallel 'example\_scalar\_function' in place on every  
// element of h\_data.**

**void example(float\* h\_data, int data\_size) {  
  
 // Allocate memory on the device (GPU), and copy the data there.  
 float\* d\_data;  
 cudaMalloc((void \*\*)&d\_data, data\_size\*sizeof(float));  
 cudaMemcpy(d\_data, h\_data, data\_size\*sizeof(float), cudaMemcpyHostToDevice);**

**// Call kernel function.  
 // This will create many threads and call ‘example\_kernel’ in each thread.  
 // In this example we create one thread for each element in the data array.  
 int num\_threads = data\_size;  
 my\_kernel<<<1, num\_threads>>>(d\_data);**

**// Copy back results from device memory to host memory, and free device memory.  
 cudaMemcpy(h\_data, d\_data, data\_size\*sizeof(float), cudaMemcpyDeviceToHost);  
 cudaFree(d\_data);**

**// CUDA exit - needed to flush printf write buffer.  
 cudaDeviceReset();**

**}**