GPGPU hardware accelerates computing by running operations in parallel across massive numbers of cores in the GPU. However, unlike thread based parallelism on traditional CPUs, GPUs have more stringent memory requirements, different levels of latency in moving memory back and forth between the GPU and main memory, and most importantly, a requirement that many threads be doing the identical thing.

The cores in a GPU are arranged in batches, called warps, of tightly linked cores that will all follow the same instruction set. These cores in a warp are bound to work in lockstep, and if one core within this set does something different, all other cores within the warp will have to pause execution until once again all cores are working in lockstep.

Branching, or conditional operation in which two possible sets of instructions branch at an if statement, can destroy parallel efficiency in a GPU. It is to be avoided at all costs--with the exception being those conditions required for thread control (e.g. a single while or if statement to determine which piece of memory a given thread is working on).

Any if or while statement other than that which is used to rewrite traditional loop structure should be examined in detail.

The exercise to be presented to the students is a kernel which performs a simple array operation. The code for this example is a derivation of an example at http://gpuray.blogspot.com/2009/07/cuda-warps-and-branching.html, which may contain more useful information. This may be presented as lecture material by the instructor as a simple demonstration, or it could be modified so that students pass an argument which determines a set size for which identical instructions will be performed (designed to range from 8 to 32 in the example given).