SOFT353 - Divergence and shared memory

Date: 20-10-16

Brand Divergence

- As a general rule, try to avoid branching statements in kernel code
- makes it a lot less efficient in some cases
- When is linked to threads organisation on the GPU
  - Crucially, all the threads in a warp share the same instruction pointer
- All threads have to be on the same instruction due to having the same instruction pointer
- If statement can split the threads, one thread's value may go inside the IF and one may go outside
  - The threads who split are deactivated
- And does the same for the code outside
- **The code has to run serially**
- The code has to be executed no matter how many threads actually use it
  - If the computation is heavey, then it becomes slow
- Sometimes if the data can be organised so that blocks execute the code they need
  - Can call if statements on that so that only the blocks do what they need instead of threads

Golden rule:

- If possible, organise blocks so that threads in the same warp follow the same path

___synchthreads

- The cause thread to pause until all of the threads in its block have reached the same ___syncthread() call
- Don't call within a statement as some threads may never reach it
  - It will cause it to block
- Split the if statement so that the ___syncthreads() is outside of if statement
  - Call the same if statement twice if needs be

Matrix Multiplication

- We're going to develop a kernel that calculates AxB, where A and B are both nxn matrices
- And how shared memory can dramatically increase the performance
- The complexity of the problem standardly is 0(nˆ3)
  - **Expensive!**
- Parallise to make faster