GPU Teaching Kit

Accelerated Computing

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

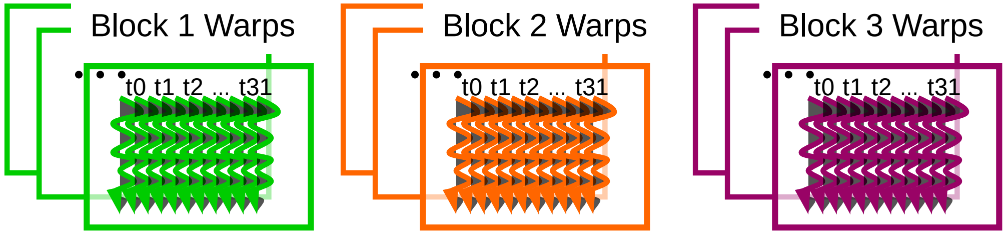
Module 5.1 – Thread Execusion Efficiency

Warps and SIMD Hardware

Objective

- To understand how CUDA threads execute on SIMD Hardware
  - Warp partitioning
  - SIMD Hardware
  - Control divergence

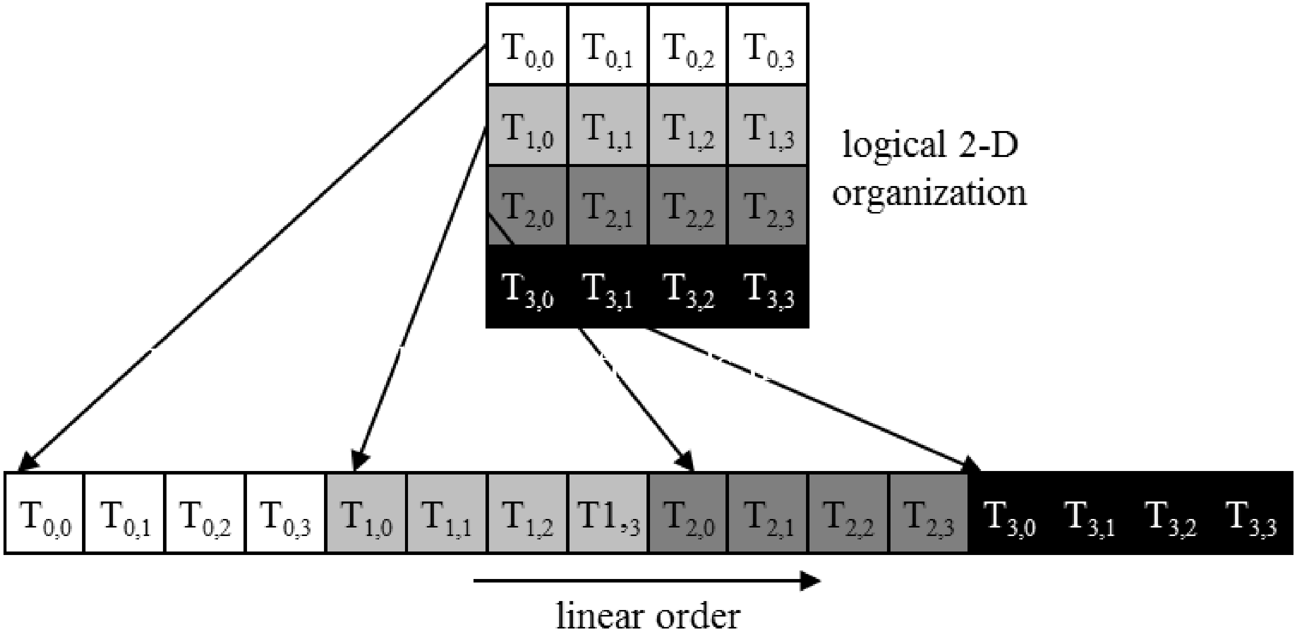
Warps as Scheduling Units



- Each block is divided into 32-thread warps
  - An implementation technique, not part of the CUDA programming model
  - Warps are scheduling units in SM
  - Threads in a warp execute in Single Instruction Multiple Data (SIMD) manner
  - The number of threads in a warp may vary in future generations

Warps in Multi-dimensional Thread Blocks

- The thread blocks are first linearized into 1D in row major order
  - In x-dimension first, y-dimension next, and z-dimension last



$T_{0,0}$ $T_{0,1}$ $T_{0,2}$ $T_{0,3}$
$T_{1,0}$ $T_{1,1}$ $T_{1,2}$ $T_{1,3}$
$T_{2,0}$ $T_{2,1}$ $T_{2,2}$ $T_{2,3}$
$T_{3,0}$ $T_{3,1}$ $T_{3,2}$ $T_{3,3}$
logical 2-D organization
$T_{0,0}$ $T_{0,1}$ $T_{0,2}$ $T_{0,3}$ $T_{1,0}$ $T_{1,1}$ $T_{1,2}$ $T1_{,3}$ $T_{2,0}$ $T_{2,1}$ $T_{2,2}$ $T_{2,3}$ $T_{3,0}$ $T_{3,1}$ $T_{3,2}$ $T_{3,3}$
linear order

Blocks are partitioned after linearization

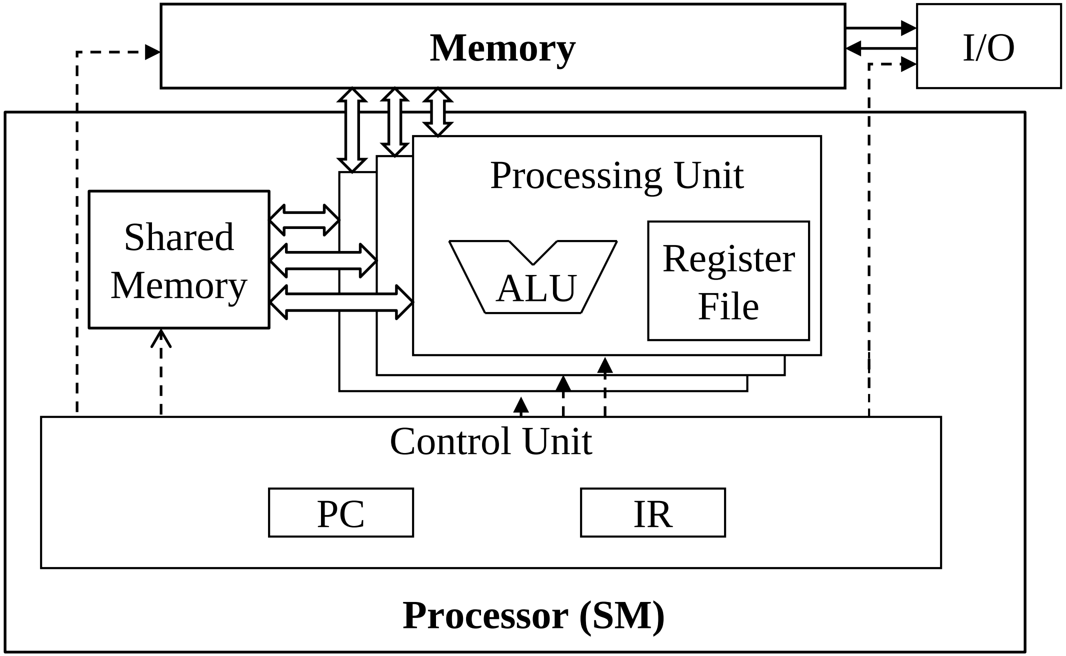
- Linearized thread blocks are partitioned
  - Thread indices within a warp are consecutive and increasing
  - Warp 0 starts with Thread 0

- Partitioning scheme is consistent across devices
  - Thus you can use this knowledge in control flow
  - However, the exact size of warps may change from generation to generation

- DO NOT rely on any ordering within or between warps
  - If there are any dependencies between threads, you must __syncthreads() to get correct results (more later).

SMs are SIMD Processors

- Control unit for instruction fetch, decode, and control is shared among multiple processing units
  - Control overhead is minimized (Module 1)



SIMD Execution Among Threads in a Warp

- All threads in a warp must execute the same instruction at any point in time

- This works efficiently if all threads follow the same control flow path
  - All if-then-else statements make the same decision
  - All loops iterate the same number of times

Control Divergence

- Control divergence occurs when threads in a warp take different control flow paths by making different control decisions
  - Some take the then-path and others take the else-path of an if-statement
  - Some threads take different number of loop iterations than others

- The execution of threads taking different paths are serialized in current GPUs
  - The control paths taken by the threads in a warp are traversed one at a time until there is no more.
  - During the execution of each path, all threads taking that path will be executed in parallel
  - The number of different paths can be large when considering nested control flow statements

Control Divergence Examples

- Divergence can arise when branch or loop condition is a function of thread indices
- Example kernel statement with divergence:
  - if (threadIdx.x > 2) { }
  - This creates two different control paths for threads in a block
  - Decision granularity < warp size; threads 0, 1 and 2 follow different path than the rest of the threads in the first warp
- Example without divergence:
  - If (blockIdx.x > 2) { }
  - Decision granularity is a multiple of blocks size; all threads in any given warp follow the same path

Example: Vector Addition Kernel

Device Code

```
// Compute vector sum C = A + B
// Each thread performs one pair-wise addition

__global__
void vecAddKernel(float* A, float* B, float* C,
  int n)
{
  int i = threadIdx.x + blockDim.x * blockIdx.x;
    if(i<n) C[i] = A[i] + B[i];
}
```

Analysis for vector size of 1,000 elements

- Assume that block size is 256 threads
  - 8 warps in each block
- All threads in Blocks 0, 1, and 2 are within valid range
  - i values from 0 to 767
  - There are 24 warps in these three blocks, none will have control divergence
- Most warps in Block 3 will not control divergence
  - Threads in the warps 0-6 are all within valid range, thus no control divergence
- One warp in Block 3 will have control divergence
  - Threads with i values 992-999 will all be within valid range
  - Threads with i values of 1000-1023 will be outside valid range
- Effect of serialization on control divergence will be small
  - 1 out of 32 warps has control divergence
  - The impact on performance will likely be less than 3%

GPU Teaching Kit

Accelerated Computing