Lesson 1.4

Introduction to CUDA
- Data Parallelism and Threads

Wen-mei Hwu - University of Illinois at Urbana-Champaign
Objective

- To learn about data parallelism and the basic features of CUDA C, a heterogeneous parallel programming interface that enables exploitation of data parallelism
  - Hierarchical thread organization
  - Main interfaces for launching parallel execution
  - Thread index to data index mapping
Data Parallelism - Vector Addition Example

vector A

\[ \begin{align*} 
& \quad \vdots & \quad \vdots \\
& \quad \vdots & \quad \vdots \\
& \quad A[N-1] \\
\end{align*} \]

vector B

\[ \begin{align*} 
& \quad \vdots & \quad \vdots \\
& \quad \vdots & \quad \vdots \\
& \quad B[N-1] \\
\end{align*} \]

vector C

\[ \begin{align*} 
C[0] & \quad C[1] & \quad C[2] \\
& \quad \vdots & \quad \vdots \\
& \quad \vdots & \quad \vdots \\
& \quad C[N-1] \\
\end{align*} \]
CUDA /OpenCL - Execution Model

- Heterogeneous host+device application C program
  - Serial parts in host C code
  - Parallel parts in device SPMD kernel C code

Serial Code (host)

Parallel Kernel (device)
KernelA<<< nBlk, nTid >>>(args);

Serial Code (host)

Parallel Kernel (device)
KernelB<<< nBlk, nTid >>>(args);
From Natural Language to Electrons

Compiler

<table>
<thead>
<tr>
<th>Natural Language (e.g., English)</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Level Language (C/C++...)</td>
<td>Instruction Set Architecture</td>
</tr>
<tr>
<td>Microarchitecture</td>
<td>Circuits</td>
</tr>
<tr>
<td>Circuits</td>
<td>Electrons</td>
</tr>
</tbody>
</table>

©Yale Patt and Sanjay Patel, *From bits and bytes to gates and beyond*
The ISA

• An Instruction Set Architecture (ISA) is a contract between the hardware and the software.

• As the name suggests, it is a set of instructions that the architecture (hardware) can execute.
A program at the ISA level

• A program is a set of instructions stored in memory that can be read, interpreted, and executed by the hardware.

• Program instructions operate on data stored in memory or provided by Input/Output (I/O) device.
A thread is a “virtualized” or “abstracted” Von-Neumann Processor
Arrays of Parallel Threads

- A CUDA kernel is executed by a grid (array) of threads
  - All threads in a grid run the same kernel code (SPMD)
  - Each thread has indexes that it uses to compute memory addresses and make control decisions

```c
i = blockIdx.x * blockDim.x + threadIdx.x;
C[i] = A[i] + B[i];
```
Thread Blocks: Scalable Cooperation

- Divide thread array into multiple blocks
- Threads within a block cooperate via shared memory, atomic operations and barrier synchronization
- Threads in different blocks do not interact

\[ i = blockIdx.x \times blockDim.x + threadIdx.x; \]
\[ C[i] = A[i] + B[i]; \]
• Each thread uses indices to decide what data to work on
  – blockIdx: 1D, 2D, or 3D (CUDA 4.0)
  – threadIdx: 1D, 2D, or 3D

• Simplifies memory addressing when processing multidimensional data
  – Image processing
  – Solving PDEs on volumes
  – …
To learn more, read Chapter 3.