Data Parallelism

- Data Parallel: performing a repeated operation (or chain of operation) over vectors of data.
- Conventionally expressed as a loop, but implementations can be constructed to perform loop operations as a single operation.
- Operations can be conditional on elements (see a2 assignment).
- Non-unit strides are often used (second example).

Examples of data parallelism:

\[ \forall i \in 0..n \quad a_1(i) = b_1(i) + c_1(i) \]
\[ \text{if}(b_2(i) \neq 0) \rightarrow a_2(i) = b_2(i) + 4 \]
\[ a_3(i) = b_3(i) + c_3(i + 1) \]

\[ \forall i \in 0, 2, 4 \cdots n \quad a_4(i) = b_4(i) + c_4(i) \]
Supporting Data Parallelism

Three basic solutions:

- Vector processors
- SIMD processors
- GPU processors
Incorporate vector operations and registers into the architecture.

Define vector load/store operations.

Masking can occur at load/store or during execution.
Vector Processing

- Have functional unit iterate over the vector registers

- Often can dispatch new vector operation each clock provided data dependencies can be satisfied. Can use chaining to forward intermediate results between adjacent (and in process) vector operations.

- **Convoy**: A set of vector operations that can potentially be executed together.

- **Gather-scatter**: Load/store mechanisms to read/write non-zero elements of memory.

- Often the hardware will support subdividing the vector elements (called lanes) so each element can be treated as sub-parts to be operated over (for example, working on 8, 16, or 32 bits in a 64-bit element).
Example of Vector Processing
Multiple Lanes

(a) (b)

Element group


\[ C[0] \]

\[ C[1] \]
\[ C[2] \]
\[ C[3] \]

Element group
SIMD
GPU/GPGPU

- GPUs provide multiple types of parallelism that was originally developed for processing the vectors and vector operations commonly found in graphics processing.

- Processing with GPUs is often considered a heterogeneous processing platform.

- CUDA/OpenCL: two models for programming heterogeneous systems (CUDA is specifically for GPGPU; OpenCL is more general).

- The real challenge is planning the migration of data into/out of the GPGPU. Often the GPGPU has limited memory space and feeding the beast becomes an issue.