Data Parallelism

UNIVERSITY OF Cincinnati

- 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:

$$\forall i \in 0..n$$
$$a1(i) = b1(i) + c1(i)$$
$$if(b2(i) \neq 0) \rightarrow a2(i) = b2(i) + 4$$
$$a3(i) = b3(i) + c3(i+1)$$

$$\forall i \in 0, 2, 4 \cdots n$$
$$a4(i) = b4(i) + c4(i)$$

Supporting Data Parallelism

Three basic solutions:

- Vector processors
- SIMD processors
- GPU processors

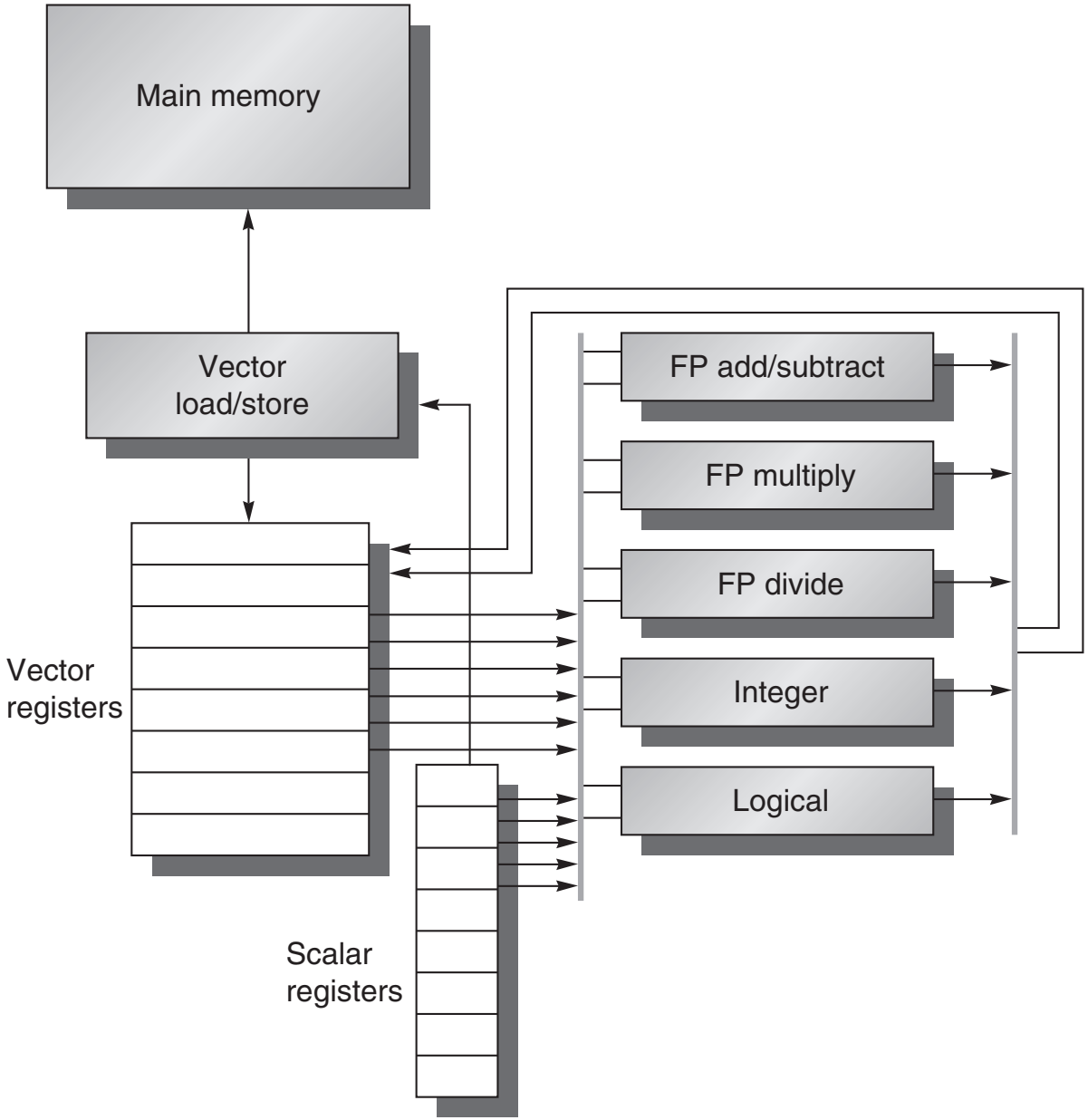
Hardware Support for Data Parallelism

- Incorporate vector operations and registers into the architecture.
- Define vector load/store operations.
- Masking can occur at load/store or during execution.

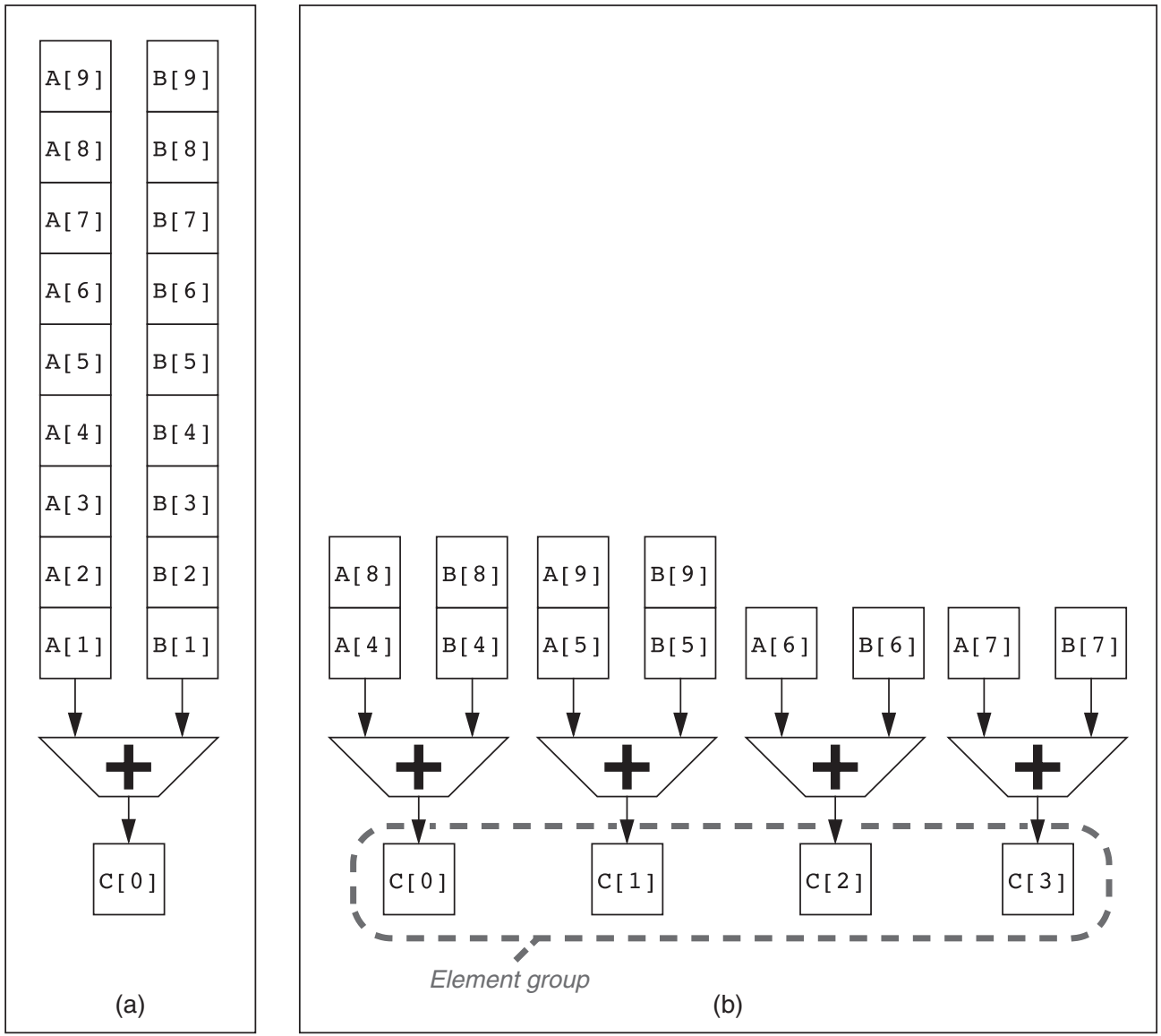
Vector Processing

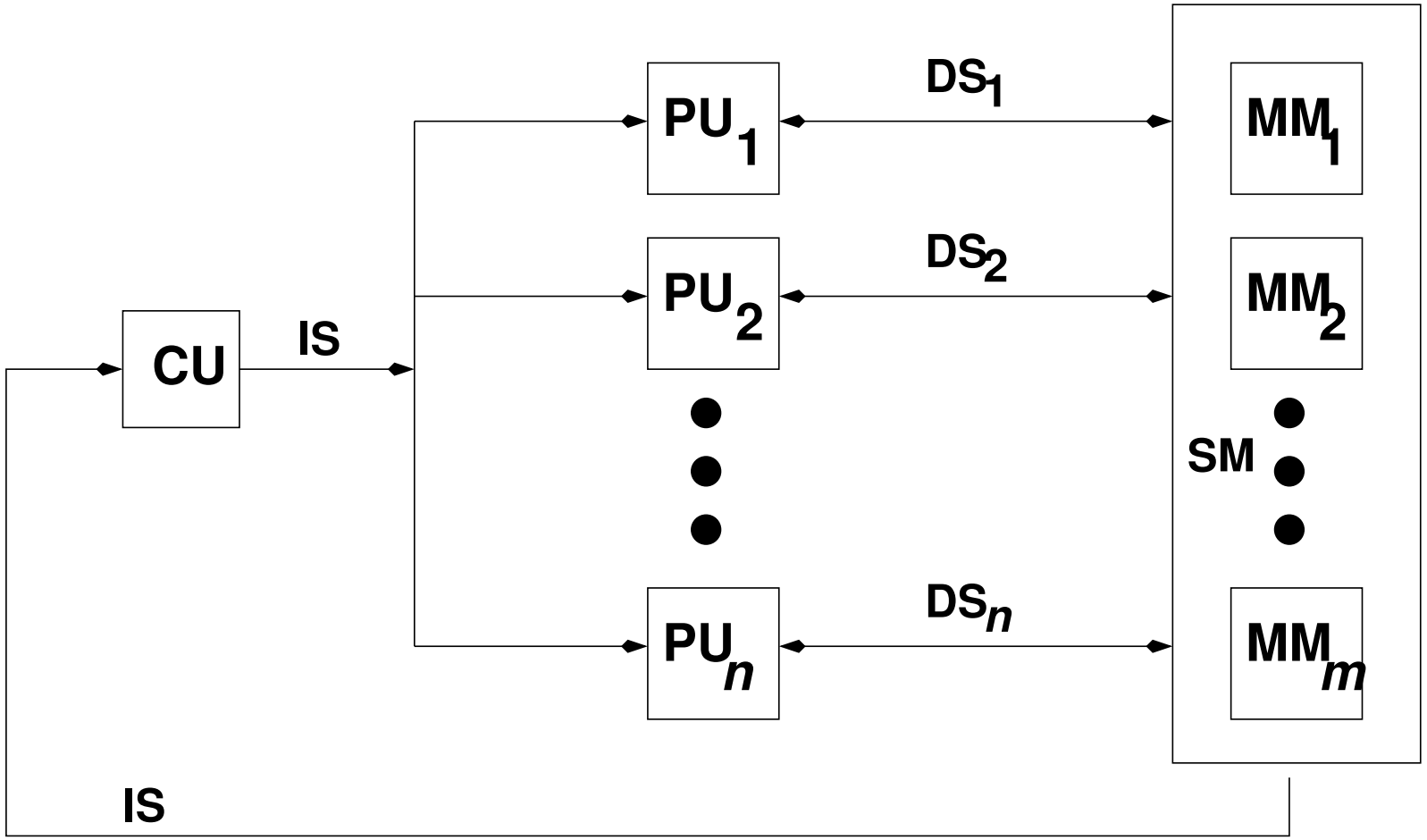
- Have functional unit iterate over the vector registers
- Often can dispatch new vector operation each clock provided data dependencies can be satisfied. Can use chaining to forward intermediate results between adjacent (and in process) vector operations.
- Convoy: A set of vector operations that can 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




CU
IS
PU1
PU2
PUn
DS1
DS2
DSn
MM1
MM2
MMm
SM
IS

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 if 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/outof the GPGPU. Often the GPGPU has limited memory space and feeding the beast becomes an issue.