Differences between Vector Architectures & GPUs

- Since both architectures are designed to execute data-level parallel programs, but take different paths, this comparison is in depth to try to gain better understanding of what is needed for DLP hardware. Fig below shows the vector term first and then the closest equivalent in a GPU.
- A SIMD Processor is like a vector processor. The multiple SIMD Processors in GPUs act as independent MIMD cores, just as many vector computers have multiple vector processors. This view would consider the NVIDIA GTX 480 as a 15-core machine with hardware support for multithreading, where each core has 16 lanes. The biggest difference is multithreading, which is fundamental to GPUs and missing from most vector processors.
- Looking at the registers in the two architectures, the VMIPS register file holds entire vectors-that is, a contiguous block of 64 doubles. In contrast, a single vector in a GPU would be distributed across the registers of all SIMD Lanes. A VMIPS processor has 8 vector registers with 64 elements, or 512 elements total. A GPU thread of SIMD instructions has up to 64 registers with 32 elements each, or 2048 elements. These extra GPU registers support multithreading.
- we assume the vector processor has four lanes and the multithreaded SIMD Processor also has four SIMD Lanes. This figure shows that the four SIMD Lanes act in concert much like a four-lane vector unit, and that a SIMD Processor acts much like a vector processor.
- In reality, there are many more lanes in GPUs, so GPU "chimes" are shorter. While a vector processor might have 2 to 8 lanes and a vector length of, say, 32-making a chime 4 to 16 clock cycles-a multithreaded SIMD Processor might have 8 or 16 lanes. A SIMD thread is 32 elements wide, so a GPU chime would just be 2 or 4 clock cycles.
- The closest GPU term to a vectorized loop is Grid, and a PTX instruction is the closest to a vector instruction since a SIMD Thread broadcasts a PTX instruction to all SIMD Lanes.
-

Type	Vector term	Closest CUDA/NVIDIA GPU term	Comment
Program abstractions	Vectorized Loop	Grid	Concepts are similar, with the GPU using the less descriptive term.
	Chime	--	Since a vector instruction (PTX Instruction) takes just two cycles on Fermi and four cycles on Tesla to complete, a chime is short in GPUs.
Machine objects	Vector Instruction	PTX Instruction	A PTX instruction of a SIMD thread is broadcast to all SIMD Lanes, so it is similar to a vector instruction.
	Gather/Scatter	Global load/store (ld.global/st.global)	All GPU loads and stores are gather and scatter, in that each SIMD Lane sends a unique address. It's up to the GPU Coalescing Unit to get unit-stride performance when addresses from the SIMD Lanes allow it.
	Mask Registers	Predicate Registers and Internal Mask Registers	Vector mask registers are explicitly part of the architectural state, while GPU mask registers are internal to the hardware. The GPU conditional hardware adds a new feature beyond predicate registers to manage masks dynamically.
Processing and memory hardware	Vector Processor	Multithreaded SIMD Processor	These are similar, but SIMD Processors tend to have many lanes, taking a few clock cycles per lane to complete a vector, while vector architectures have few lanes and take many cycles to complete a vector. They are also multithreaded where vectors usually are not.
	Control Processor	Thread Block Scheduler	The closest is the Thread Block Scheduler that assigns Thread Blocks to a multithreaded SIMD Processor. But GPUs have no scalar-vector operations and no unit-stride or strided data transfer instructions, which Control Processors often provide.
	Scalar Processor	System Processor	Because of the lack of shared memory and the high latency to communicate over a PCI bus (1000s of clock cycles), the system processor in a GPU rarely takes on the same tasks that a scalar processor does in a vector architecture.
	Vector Lane	SIMD Lane	Both are essentially functional units with registers.
	Vector Registers	SIMD Lane Registers	The equivalent of a vector register is the same register in all 32 SIMD Lanes of a multithreaded SIMD Processor running a thread of SIMD instructions. The number of registers per SIMD thread is flexible, but the maximum is 64, so the maximum number of vector registers is 64.
	Main Memory	GPU Memory	Memory for GPU versus System memory in vector case.

- With respect to memory access instructions in the two architectures, all GPU loads are gather instructions and all GPU stores are scatter instructions. The explicit unit-stride load and store instructions of vector architectures versus the implicit unit stride of GPU programming is why writing efficient GPU code requires that programmers think in terms of SIMD operations, even though the CUDA programming model looks like MIMD.
- The two architectures take very different approaches to hiding memory latency. Vector architectures amortize it across all the elements of the vector by having a deeply pipelined access so you pay the latency only once per vector load or store. Hence, vector loads and stores are like a block transfer between memory and the vector registers. In contrast, GPUs hide memory latency using multithreading
- With respect to conditional branch instructions, both architectures implement them using mask registers. Both conditional branch paths occupy time and/or space even when they do not store a result. The difference is that the vector compiler manages mask registers explicitly in software while the GPU hardware and assembler manages them implicitly using branch synchronization markers.
- The conditional branch mechanism of GPUs gracefully handles the strip-mining problem of vector architectures . When the vector length is unknown at compile time, the program must calculate the modulo of the application vector length and the maximum vector length and store it in the vector length register. The strip-minded loop then resets the vector length register to the maximum vector length for the rest of the loop. This case is simpler with GPUs since they just iterate the loop until all the SIMD Lanes reach the loop bound.
- The control processor of a vector computer plays an important role in the execution of vector instructions. It broadcasts operations to all the vector lanes and broadcasts a scalar register value for vector-scalar operations. The control processor is missing in the GPU. The closest analogy is the Thread Block Scheduler, which assigns Thread Blocks (bodies of vector loop) to multithreaded SIMD Processors.
- The scalar processor in a vector computer executes the scalar instructions of a vector program; that is, it performs operations that would be too slow

to do in the vector unit.. vector unit" in a GPU must do computations that you would expect to do on a scalar processor in a vector computer

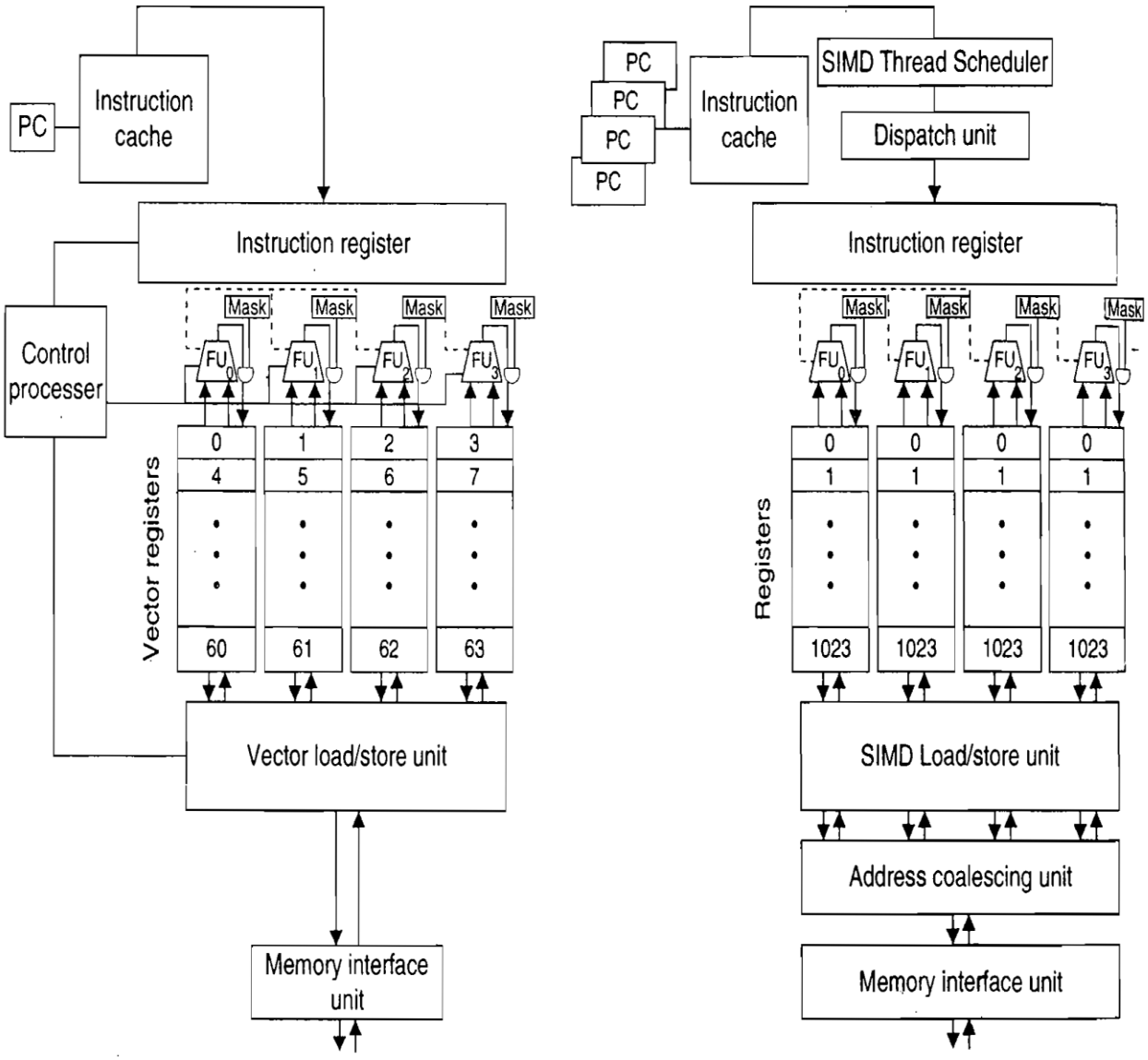

Fig: vector processor with four lanes on the left & multithreaded SIMD Processor of a GPU with four SIMD Lanes on the right.