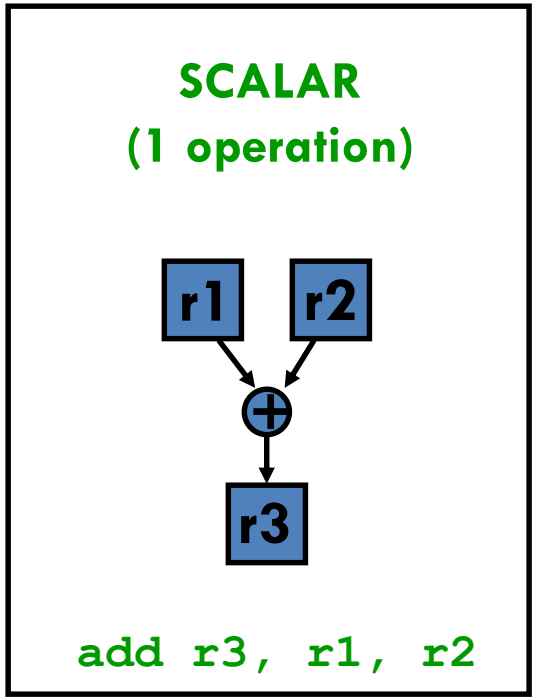
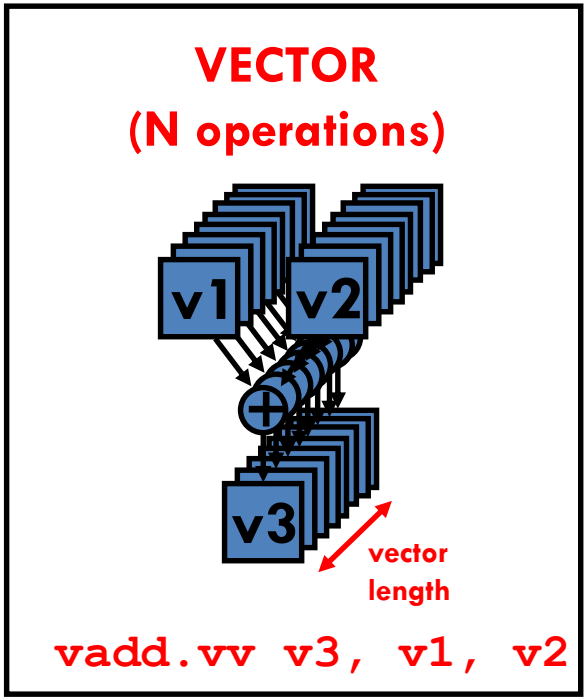
CSE 502: Computer Architecture

Data-Level Parallelism

Vector Processors



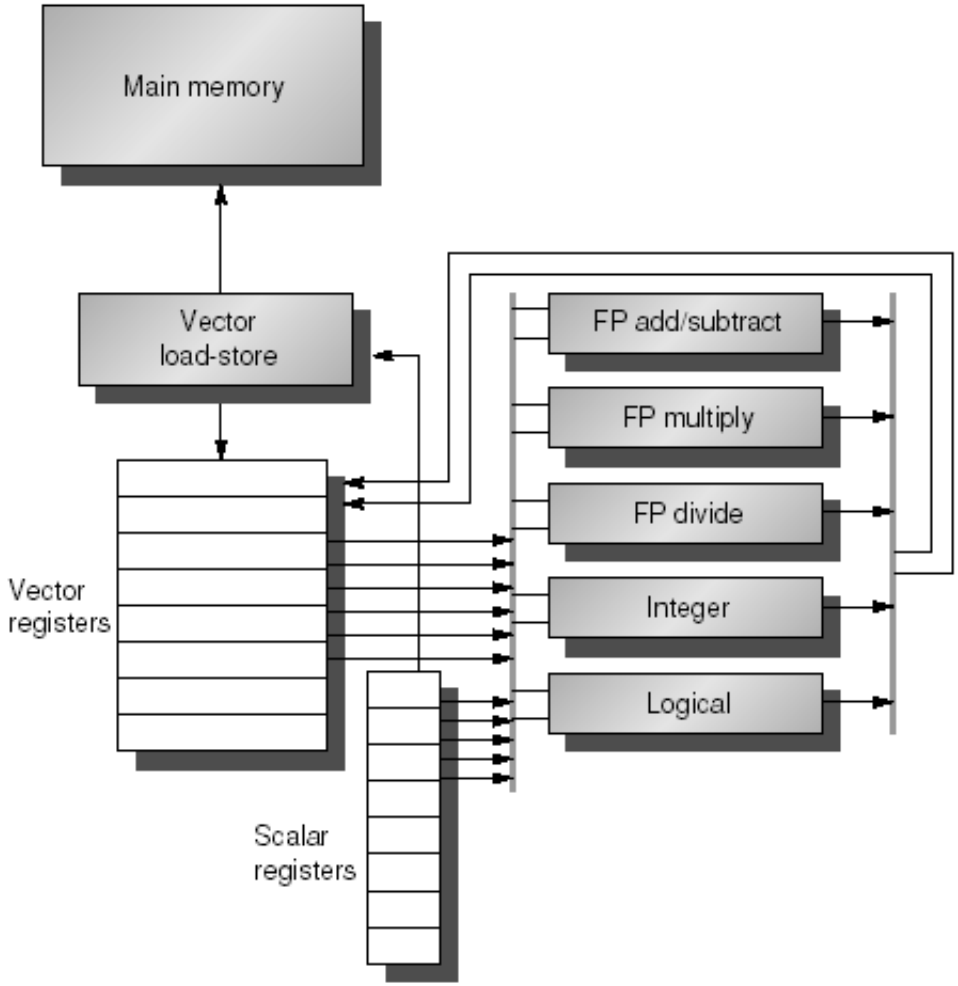


- Scalar processors operate on single numbers (scalars)
- Vector processors operate on sequences of numbers (vectors)

What's in a Vector Processor?

- A scalar processor (e.g., a MIPS processor)
  - Scalar register file (32 registers)
  - Scalar functional units (arithmetic, load/store, etc)
- A vector register file (a 2D register array)
  - Each register is an array of elements
    - e.g., 32 registers with 32 64-bit elements per register
  - MVL = maximum vector length = max # of elements per register
- A set of vector functional units
  - Integer, FP, load/store, etc
  - Sometimes vector and scalar units are combined (share ALUs)

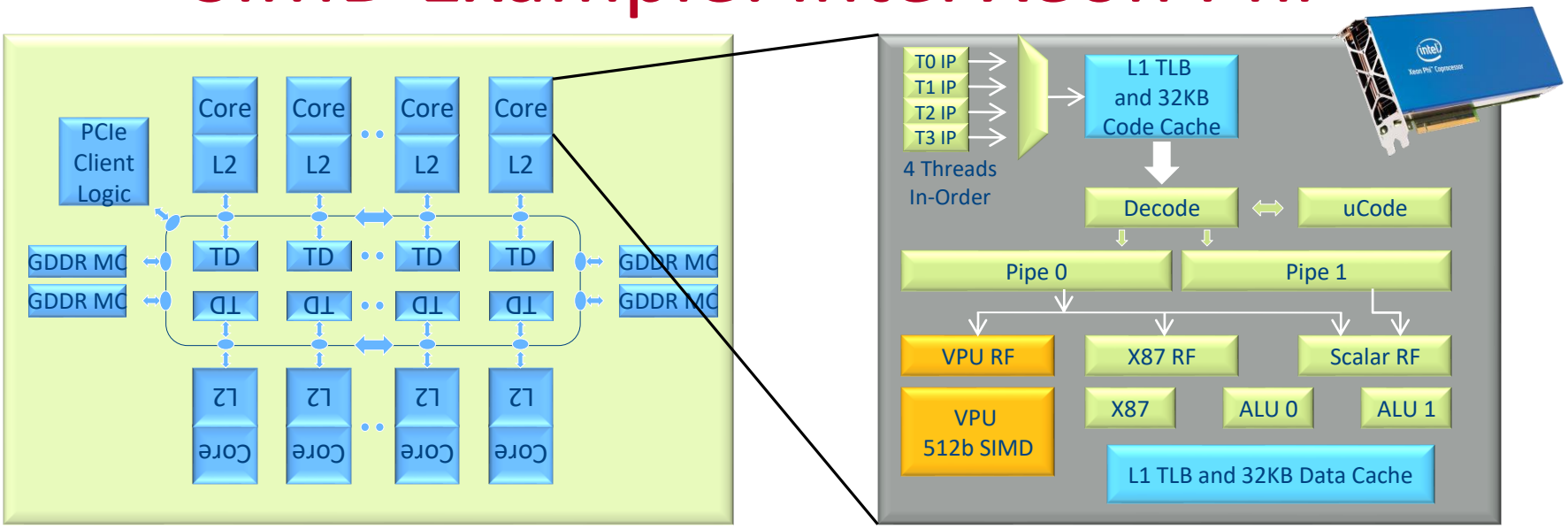
Example of Simple Vector Processor



Advantages of Vector ISAs

- Compact: single instruction defines N operations
  - Amortizes the cost of instruction fetch/decode/issue
  - Also reduces the frequency of branches
- Parallel: N operations are (data) parallel
  - No dependencies
  - No need for complex hardware to detect parallelism
  - Can execute in parallel assuming N parallel datapaths
- Expressive: memory operations describe patterns
  - Continuous or regular memory access pattern
  - Can prefetch or accelerate using wide/multi-banked memory
  - Amortizes high latency for 1st element over sequential pattern

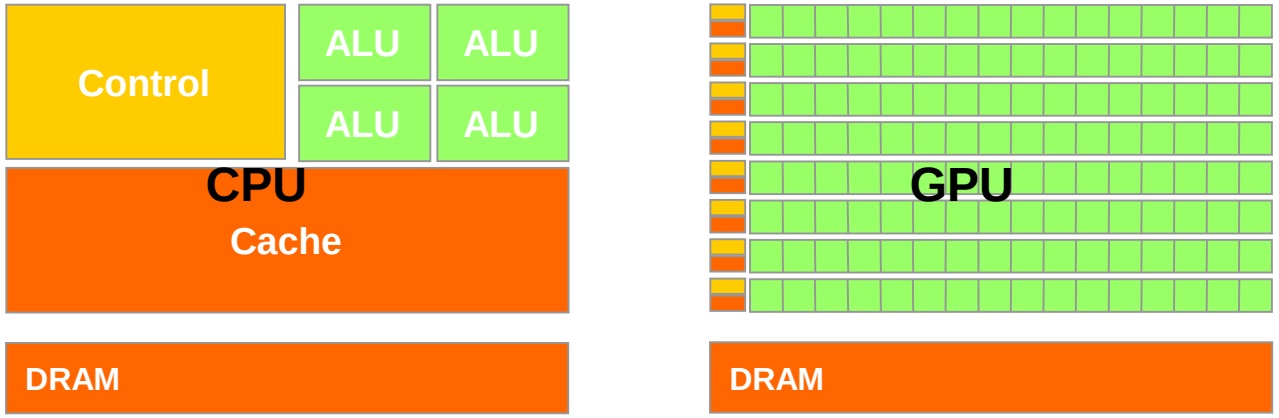
SIMD Example: Intel Xeon Phi



- Multi-core chip with Pentium-based SIMD processors
  - Targeting HPC market (Goal: high GFLOPS, GFLOPS/Watt)
- 4 hardware threads + wide SIMD units
  - Vector ISA: 32 vector registers (512b), 8 mask registers, scatter/gather
- In-order, short pipeline
  - Why in-order?

Graphics Processing Unit (GPU)

- Architecture for compute-intensive highly data-parallel computation
  - Exactly what graphics rendering needs
  - Transistors devoted to data processing
    - Not caching and flow control



Data Parallelism in GPUs

- GPUs use massive DLP to provide high FLOPs
  - More than 1 Tera DP FLOP in NVIDIA GK110
- SIMT execution model
  - Single instruction multiple threads
  - Trying to distinguish itself from both “vectors” and “SIMD”
  - A key difference: better support for conditional control flow
- Program it with CUDA or OpenCL (among other things)
  - Extensions to C
  - Run “shader task”(snippet of scalar code) on many elements
  - Internally, GPU uses scatter/gather and vector-mask-like ops

GPGPU (General Purpose GPU)

- In the beginning…
  - Originally could only perform "shader" computations on images
  - So, programmers started using this framework for computation
  - Puzzle to work around the limitations, unlock the raw potential
- As GPU designers notice this trend…
  - Hardware provided more "hooks" for computation
  - Provided some limited software tools
- GPU designs are now fully embracing compute
  - More programmability features in each generation
  - Industrial-strength tools, documentation, tutorials, etc.
  - Can be used for in-game physics, etc.
  - A major initiative to push GPUs beyond graphics (HPC)