SIMD hardware

Terminology note

SIMD is an umbrella term describing architectures designed for DLP

Units that use parallelism to apply the same operation to different data

Vectors are an abstraction that defines data layout

SIMD architectures represent/understand these in different ways

Marketing has made this a bit fuzzy

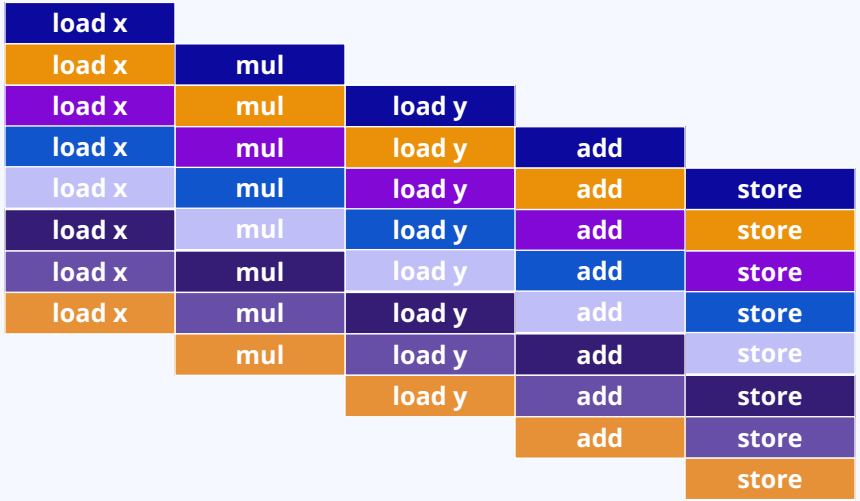
Vector processors: classic supercomputers (heyday in the 80s) that operate on large vectors of data using SIMD principles

SIMD units: hardware in modern CPUs that computes on small vectors

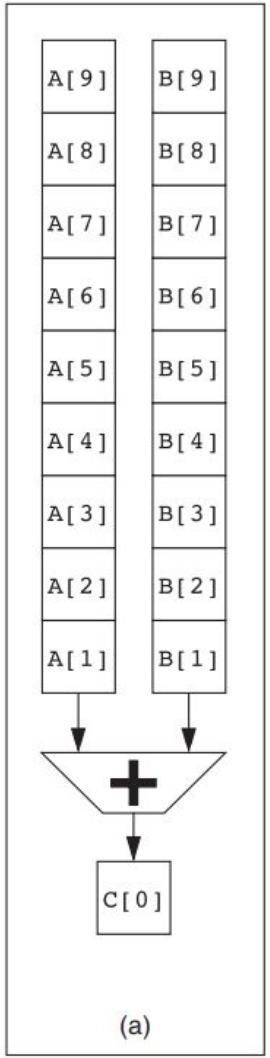
Today: focusing on SIMD **hardware**

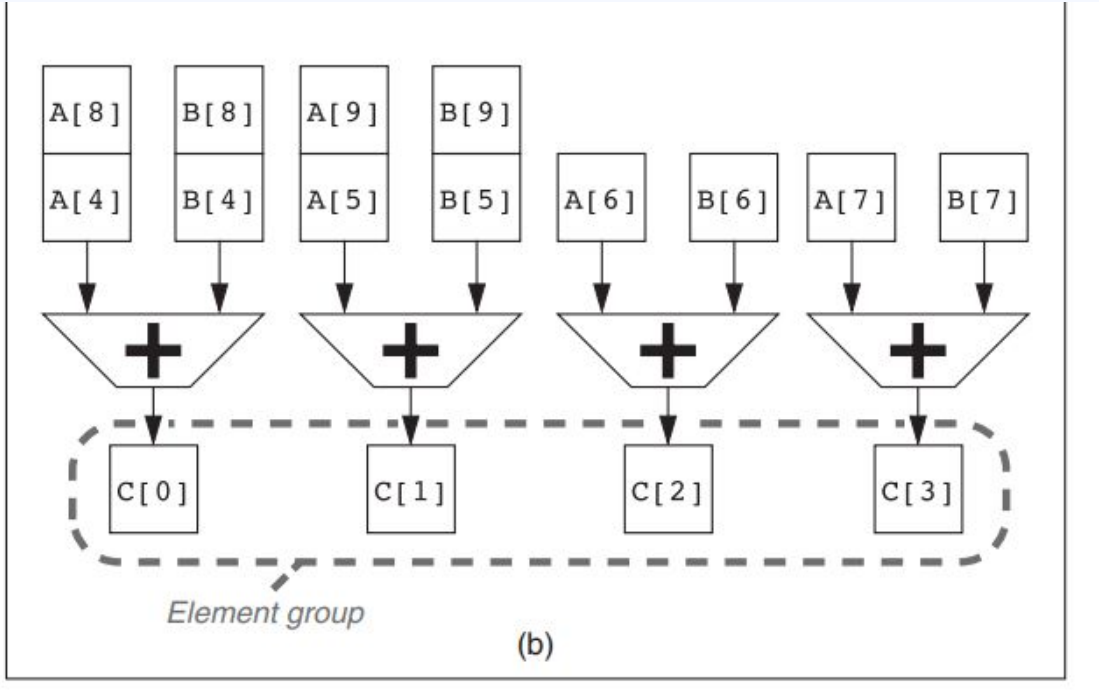
Hardware tradeoffs?

load x	load x	load x	load x	load x	load x	load x	load x
mul	mul	mul	mul	mul	mul	mul	mul
load y	load y	load y	load y	load y	load y	load y	load y
add	add	add	add	add	add	add	add
store	store	store	store	store	store	store	store

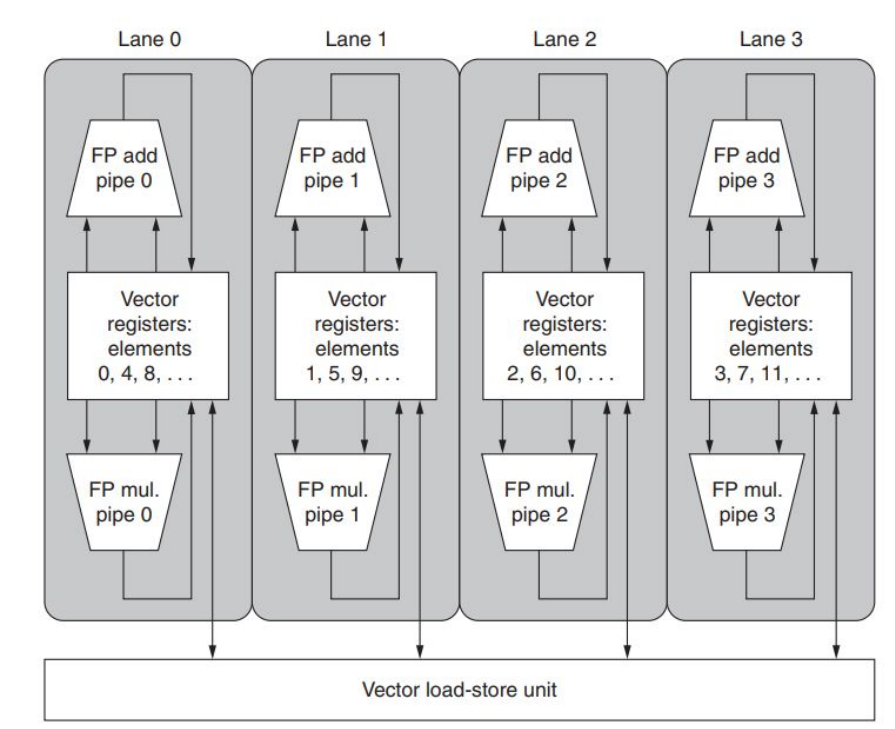


Vector processor functional units: two approaches





Vector processor lanes



What does this imply about the memory controller?

Cray-1 Architecture (1976)

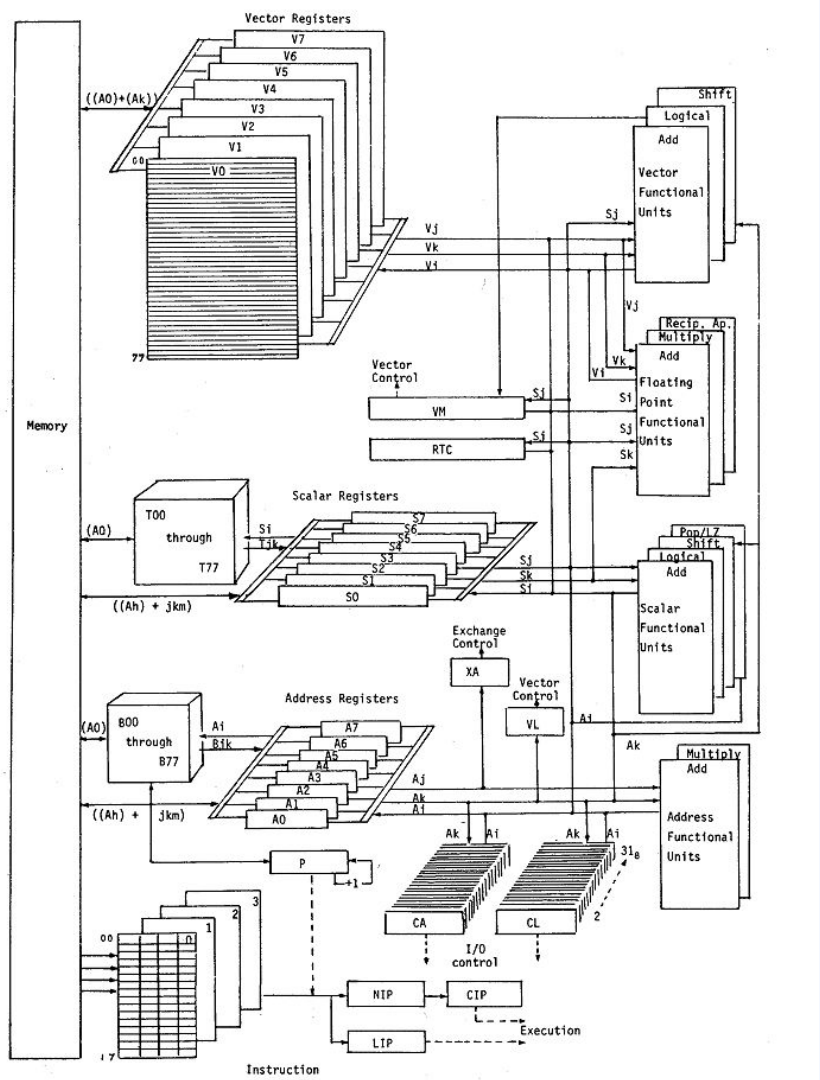


image source

Cray X1 architecture (2003)

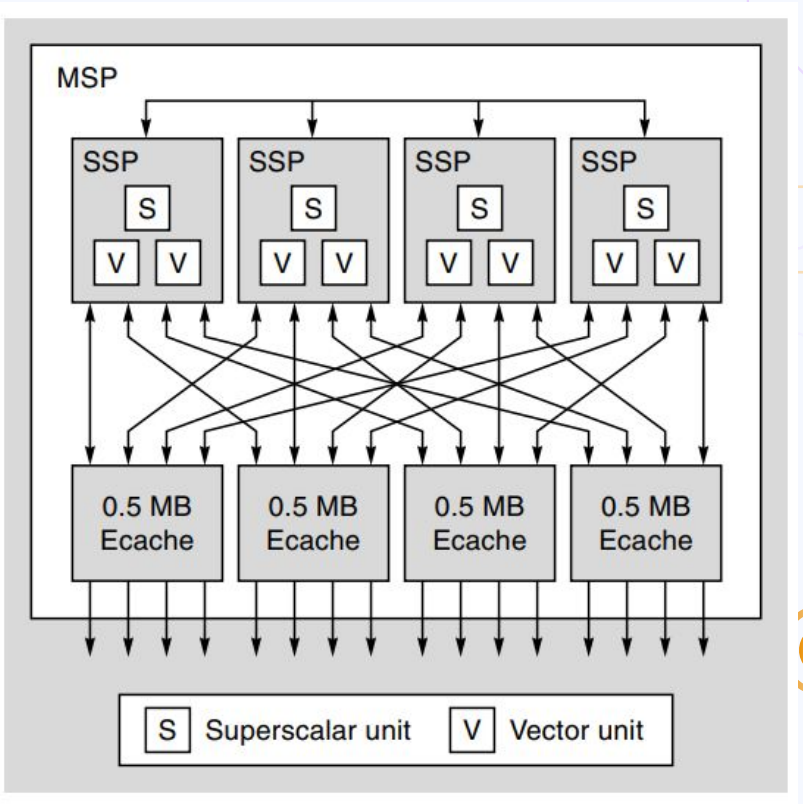
ISA designed from scratch

Multi-stream processor consisting of four single-stream processors

- Each SSP has: scalar unit/scalar cache, 2-lane vector unit

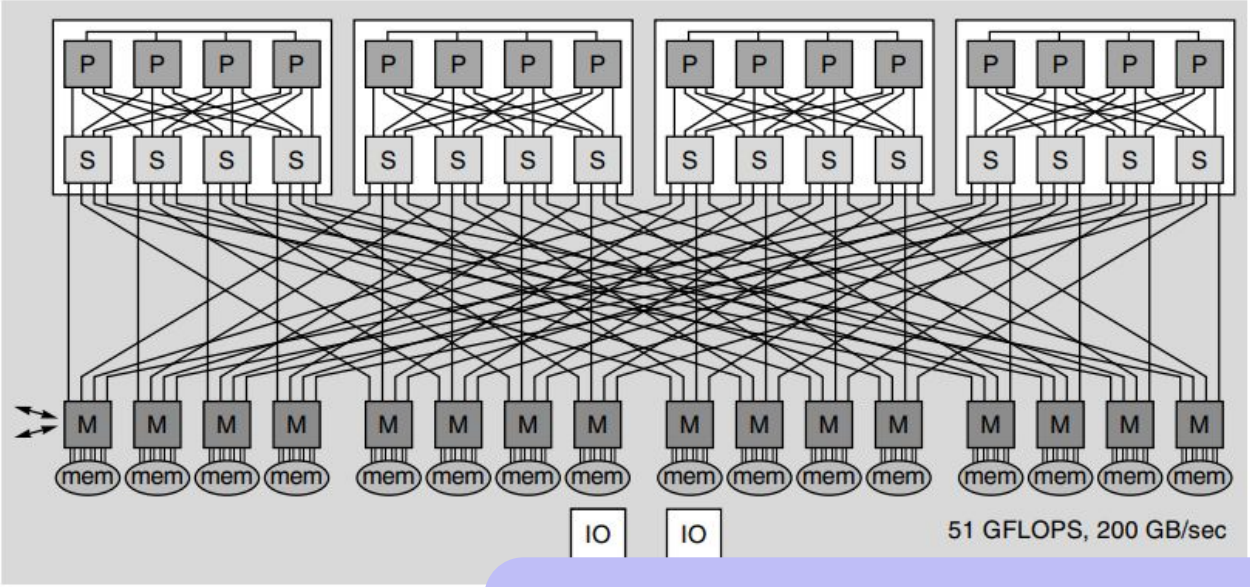
Connected to external caches (mostly for scalars, but can be used by vectors for programs w/ high temporal locality, or bypassed)

Each MSP can have up to 2048 outstanding memory requests



H&P fig. G.11

Cray X1 nodes (H&P fig. G.12)

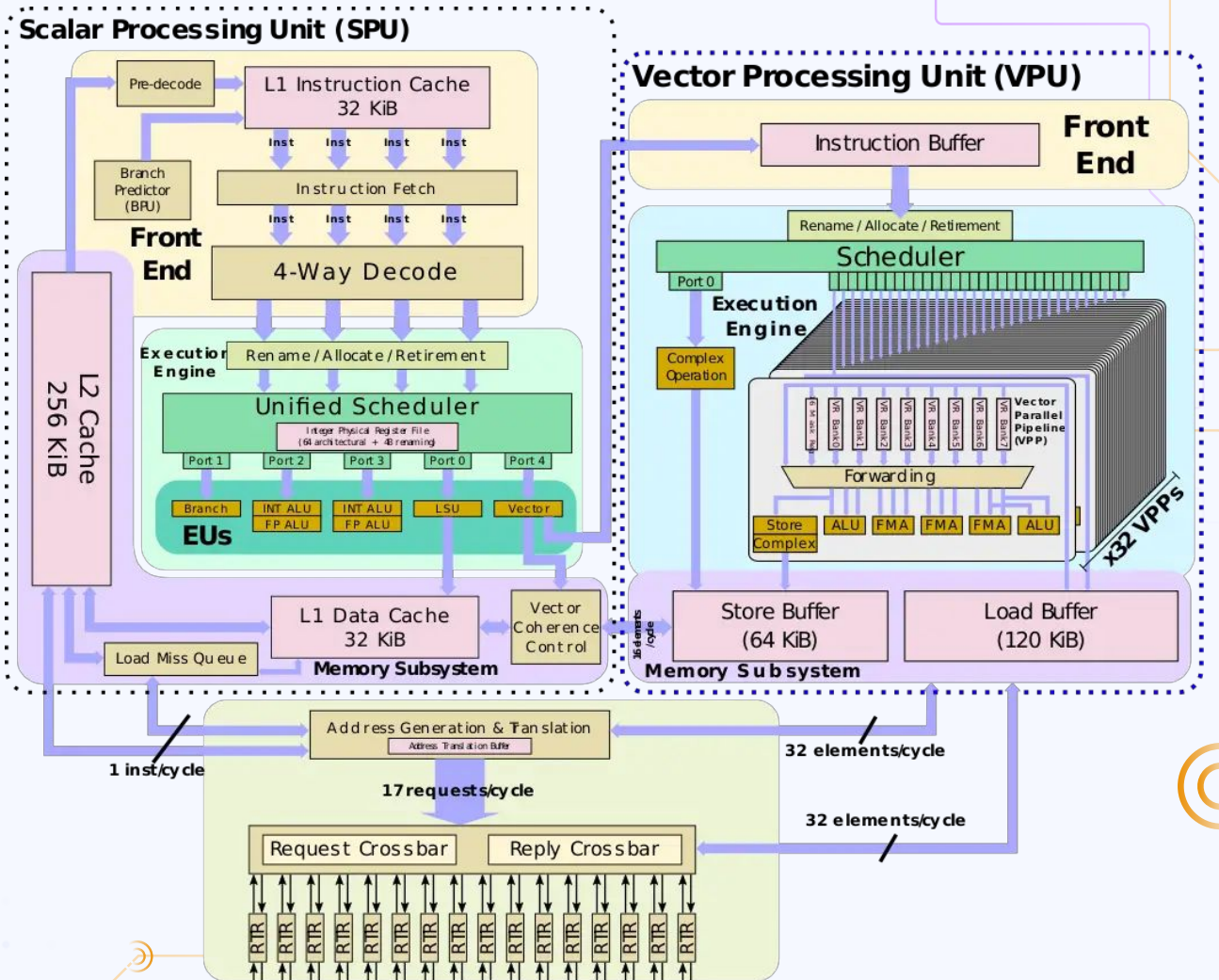


NUMA: non-uniform memory access
ecaches only cache their local memory
writes to remote memory invalidate
corresponding ecache data

NEC SX Aurora (2018)

image source

Shared LLC (last-level cache) with 128 banks
Uses high-bandwidth memory (HbM)



What are the limits of vector processors?

VLIW paper

Fisher, Joseph A. "Very long instruction word architectures and the ELI-512." Proceedings of the 10th annual international symposium on Computer architecture. 1983. (link)

WHY NOT VECTOR MACHINES?

Vector machines seem to offer much more parallelism than the factor of 2 or 3 that current VLIWs offer. Although vector machines have their place, we don't believe they have much chance of success on general-purpose scientific code. They are crucifyingly difficult to program, and they speed up only inner loops, not the rest of the code.

And vectorizing works only on inner loops; the rest of the code gets no speedup whatsoever. Even if 90% of the code were in inner loops, the other 10% would run at the same speed as on a sequential machine. Even if you could get the 90% to run in zero time, the other 10% would limit the speedup to a factor of 10.

Amdahl's law

Used to assess theoretical effectiveness of speedup

In a nutshell: gains in speeding up a portion of a program are limited by the fraction of time that portion is actually used

Mathematically:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

For parallelization: serial bottleneck (non-parallelizable code) limits effectiveness of vector processors

Supercomputer evolution

Cray, but not vector (massive multiprocessor instead)

NSF National Center for Atmospheric Research
@NCAR_Science · **Follow**

source

44 yrs after our CRAY-1A #supercomputer was lowered into a computing chamber in the basement of our Mesa Lab, the venerable machine was finally lifted back out.

This week, it took a trip north, where it will be displayed in the lobby of the NCAR-Wyoming Supercomputing Center

NOAA completes upgrade to weather and climate supercomputer system

Increased computing power will enable a significant upgrade to the 'American' forecast model

Focus areas: Weather **Topics:** supercomputing/IT **Share:**

August 10, 2023

source

NVIDIA DEVELOPER Home Blog Forums Docs Downloads Training

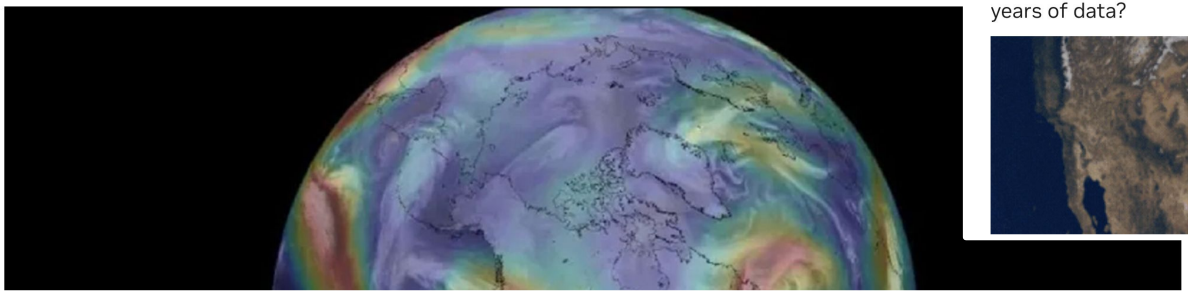
Technical Blog Search blog

Hidden GEM: Canadian Weather Forecasts to Run on NVIDIA-System

June 29, 2021 by Gilad Shainer

2 mins 0 Comments Share

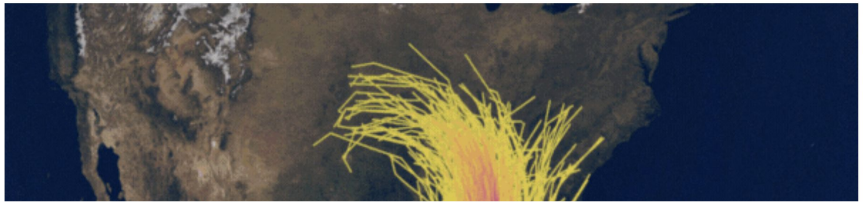
source



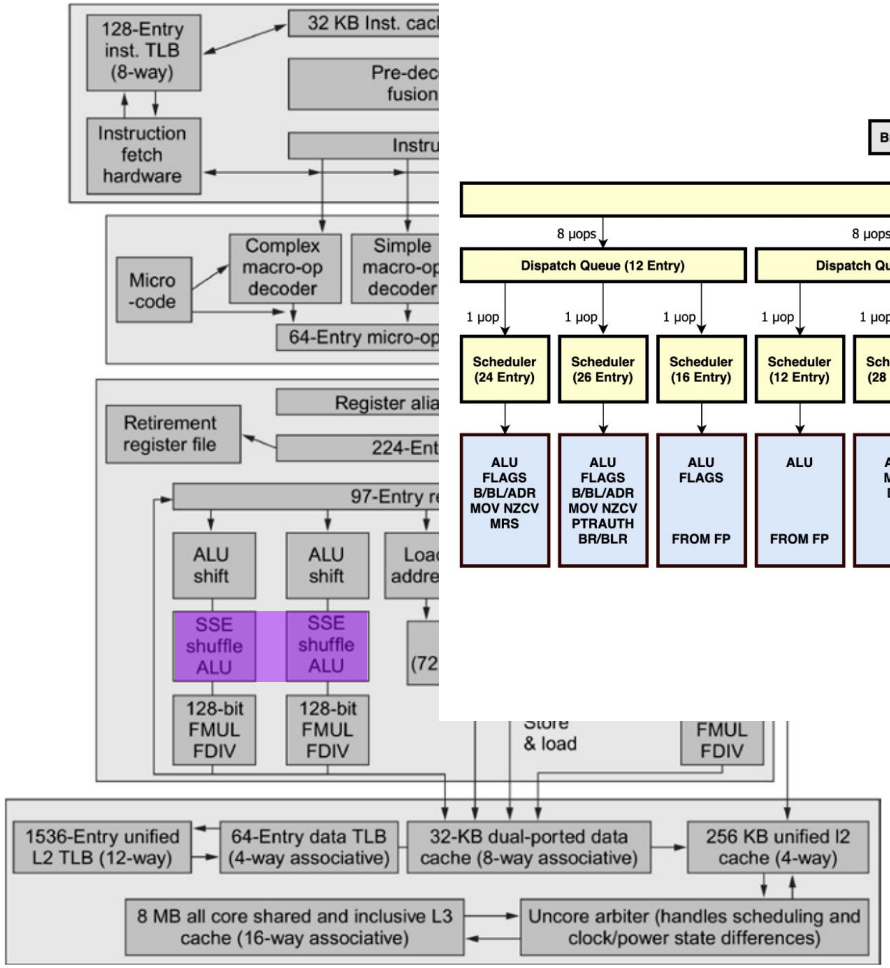
Predict Extreme Weather Events in Minutes Without a Supercomputer

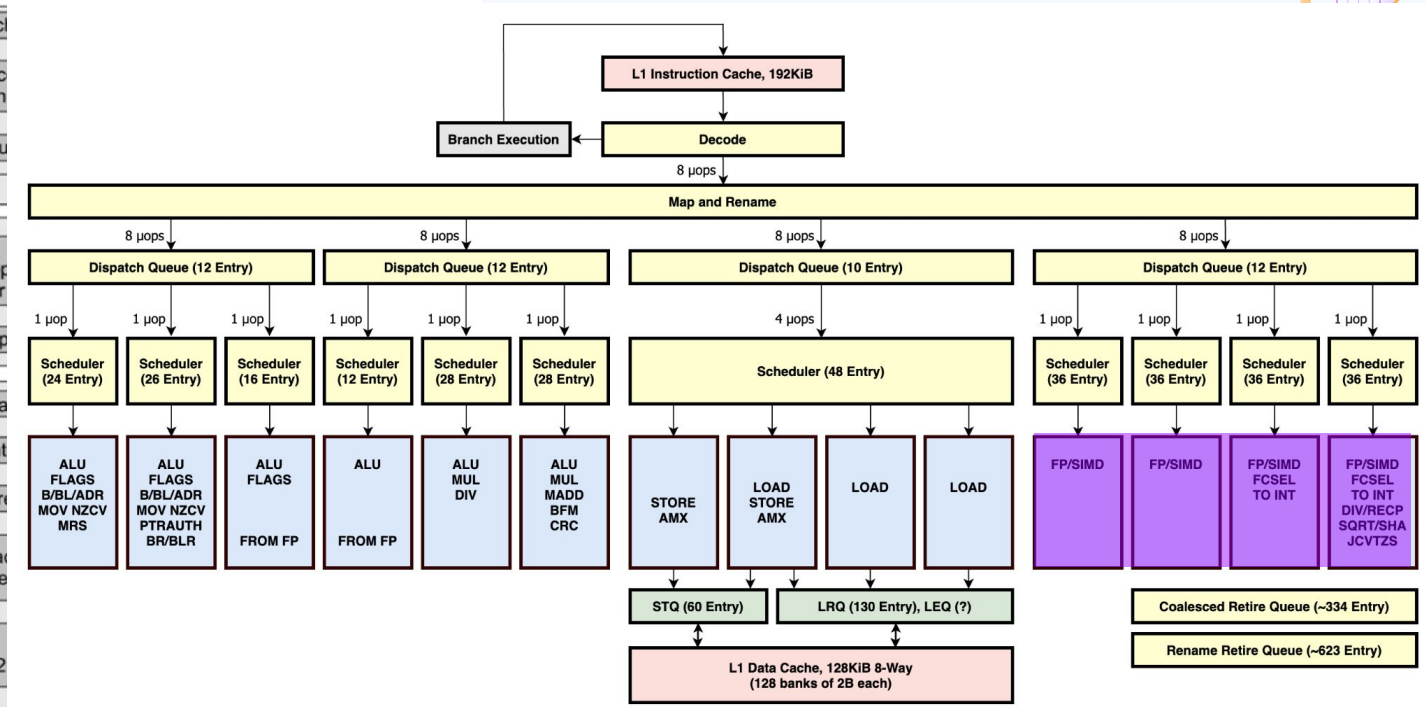
How can you generate climate forecasts in minutes using the HENS model and 27,000 years of data?

source



What about DLP for the rest of us?





(from "OOO IRL" lecture)

Accelerate

The Accelerate framework provides high-performance, energy-efficient computation on the CPU by leveraging its vector-processing capability. Accelerate performs optimized large-scale mathematical computations and image calculations so you can write apps that leverage machine learning, data compression, signal processing, and more.

source

SIMD for multimedia

RGBA images: 8 bits/channel (32 bits total)

Audio: 8, 16, 24, or 32 bits per sample

Simplifications of SIMD for multimedia: might not need strided access, gather/scatter, masked operations, custom vector length

→ Doesn't typically make sense to put a powerful VPU on a processor

Enter multimedia SIMD extensions

How can smaller data widths make SIMD functionality easier to add to CPUs?

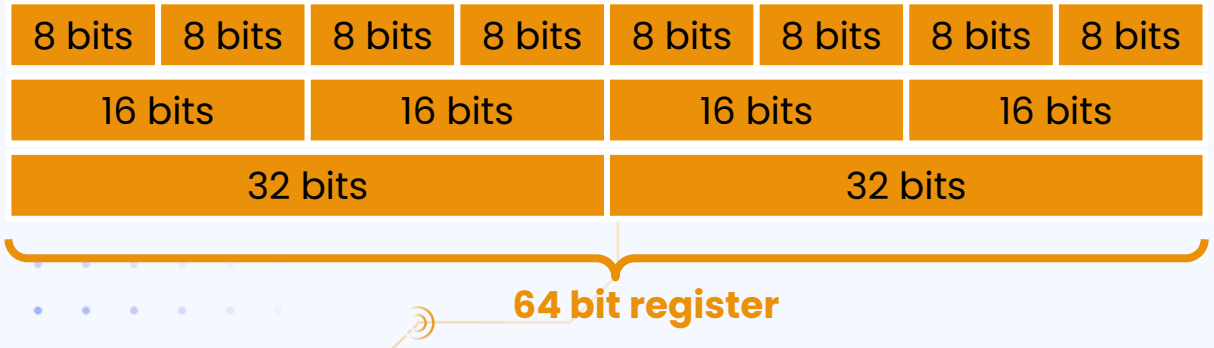
RISC-V P: packed SIMD

(Still a draft extension proposal)

Reuses floating-point registers as packed “vector” registers

Packs multiple values in one register based on configuration

- Can use existing loads/stores (*where does this get tricky?*)
- Requires vectorization support in FUs *(how is a vectorized add different from a 64-bit add?)*



ARMv6 SIMD

Packs multiple 16- or 8-bit values into 32 bit registers

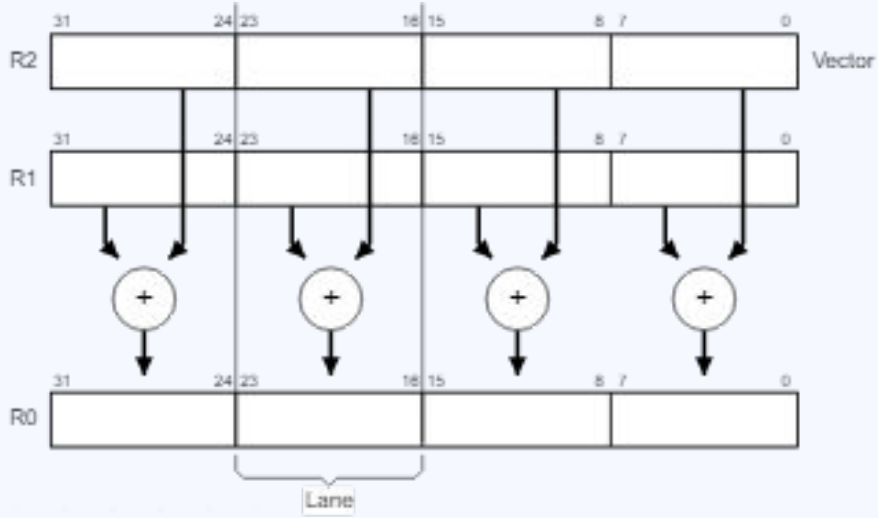


image source

Note: later ARM chips use NEON (their “Advanced SIMD” extension), storing vectors in 64- and 128-bit registers

x86: MMX, SSE, AVX

MMX: not an acronym, packs values in 64-bit registers, supports integer operations only

SSE: “Streaming SIMD Extensions”, 128-bit registers, allows for floating point

AVX: “Advanced Vector Extensions”, 8x32 or 4x64 vector registers (AVX 2 adds gather, AVX 512 supports 512-bit registers)

In typical x86 fashion, operand size is fixed in the opcode (so there are hundreds of instructions for each extension)

x86 AVX-512 VNNI

Vector Neural Network Instructions
Useful for CNNs (Convolutional Neural Networks)

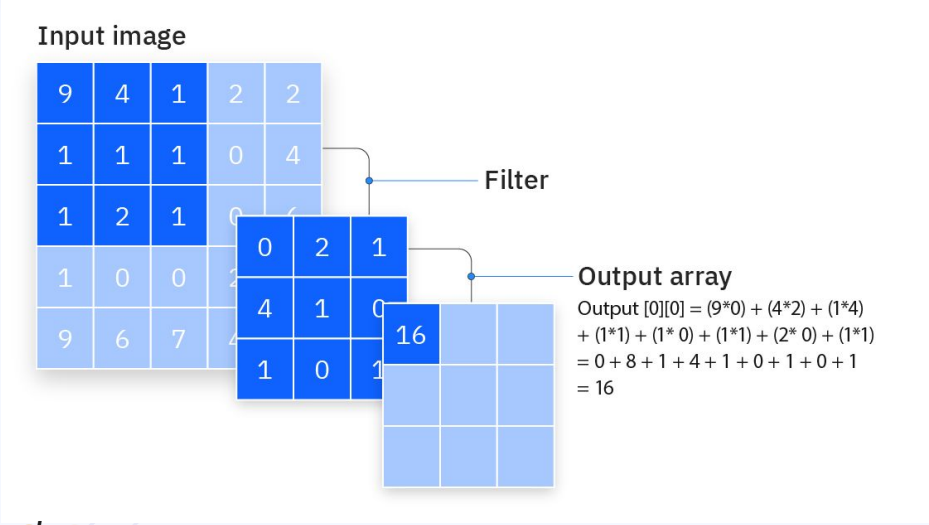


Image source

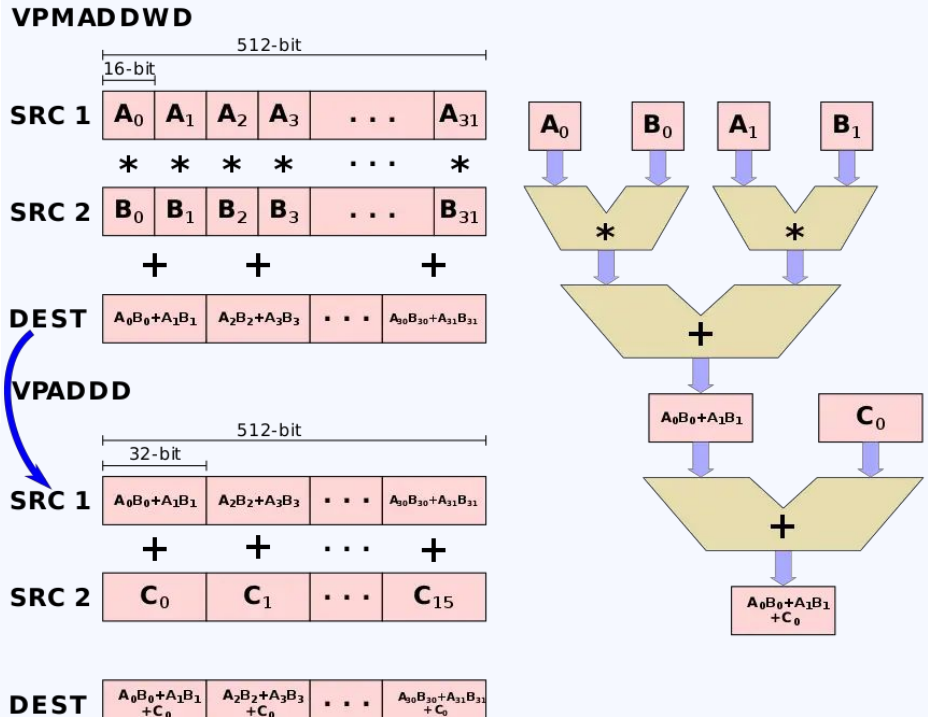


Image source

From the Intel optimization manual

P 5-11 (193): Converting to SIMD chart

P 8-9 (287): Blocking (handling large matrices)

P 14-2 (390): PCMPxSTRy (see also 14-12 onward)

P 15-7 (445): Mixing SSE and AVX (YMM register)

P 15-20 (458): Data alignment and caches

P 15-24 (462): Masked loads and paging

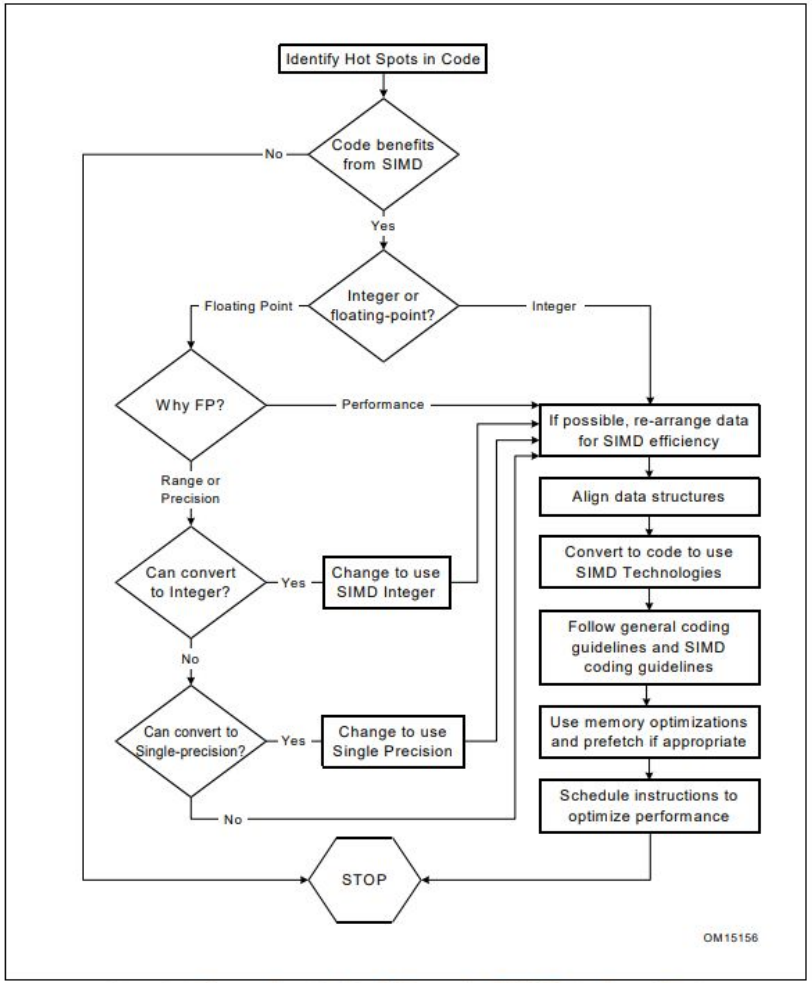


Figure 5-3. Converting to Intel® Streaming SIMD Extensions Chart

Extra slides: Vector processor performance

Measuring performance: convoys and chimes

Convoy: set of vector instructions that can potentially execute together

Chime: time it takes to execute a convoy

```
vle32.v v0, s1
vmul.vx v1, v0, t0
vle32.v v2, s2
vadd.vv v3, v1, v2
vse32.v v3, t1
```

Approximation of runtime for this vector machine: 3 chimes (~3 * vlen/lanes clock cycles)
What complicates this metric?

Measuring performance: chaining

```
vle32.v v0, s1
vmul.vx v1, v0, t0
vle32.v v2, s2
vadd.vv v3, v1, v2
vse32.v v3, t1
```