
The top-left corner of the slide features a decorative graphic of circuit lines. It includes a horizontal blue line, a diagonal purple line, and several orange lines with circular nodes at their intersections, resembling a printed circuit board layout.

SIMD in modern computers

The bottom-right corner contains another decorative graphic. It features a grid of small blue dots, a diagonal line of blue dots, and several orange lines with circular nodes, similar to the top-left corner. There are also some purple geometric shapes and a small cluster of orange dots.

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
```

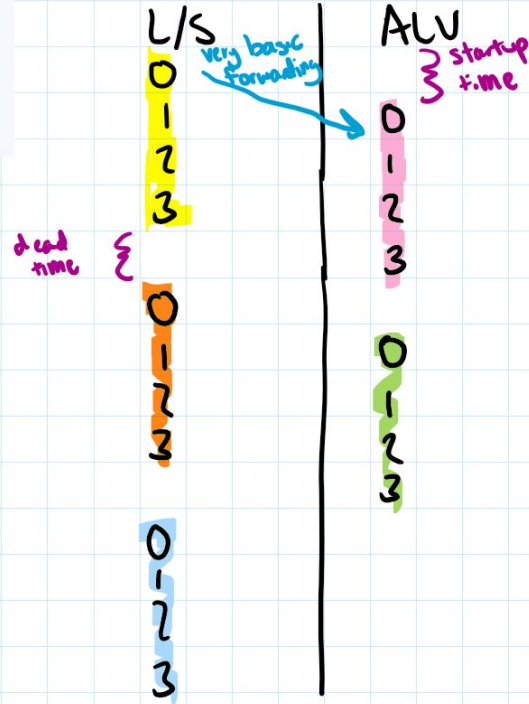
```
vle32.v v2, s2  
vadd.vv v3, v1, v2
```

```
vse32.v v3, t1
```

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

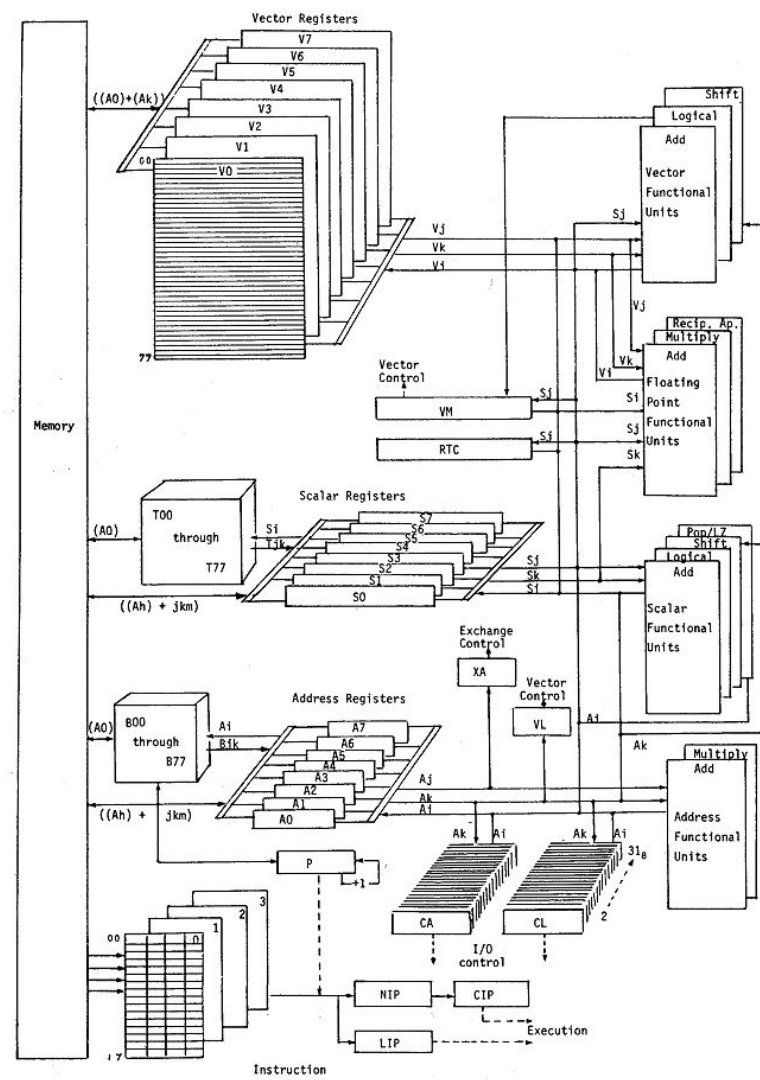
Measuring performance: chaining

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



Cray-1 Architecture (1976)

image source



Cray X1 architecture (2003)

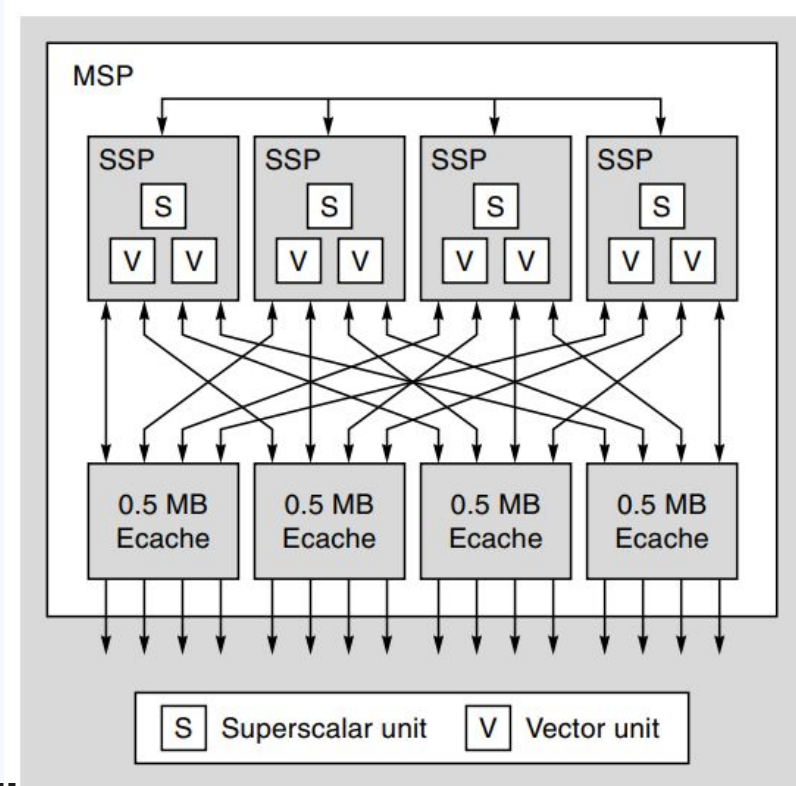
ISA designed from scratch

Multi-stream processor consisting of four single-stream processors

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

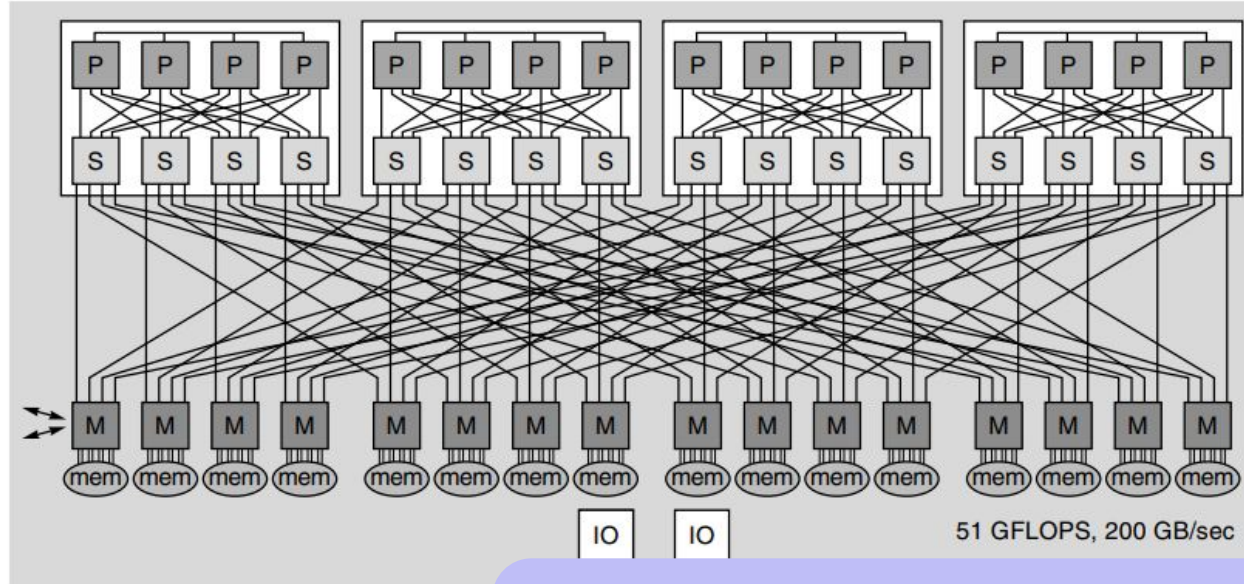
Connected to external caches (mostly for scalars, but can be used by vectors for programs w/ high temporal locality, or bypassed)

Each MSP can have up to 2048 outstanding memory requests



H&P fig. G.11

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

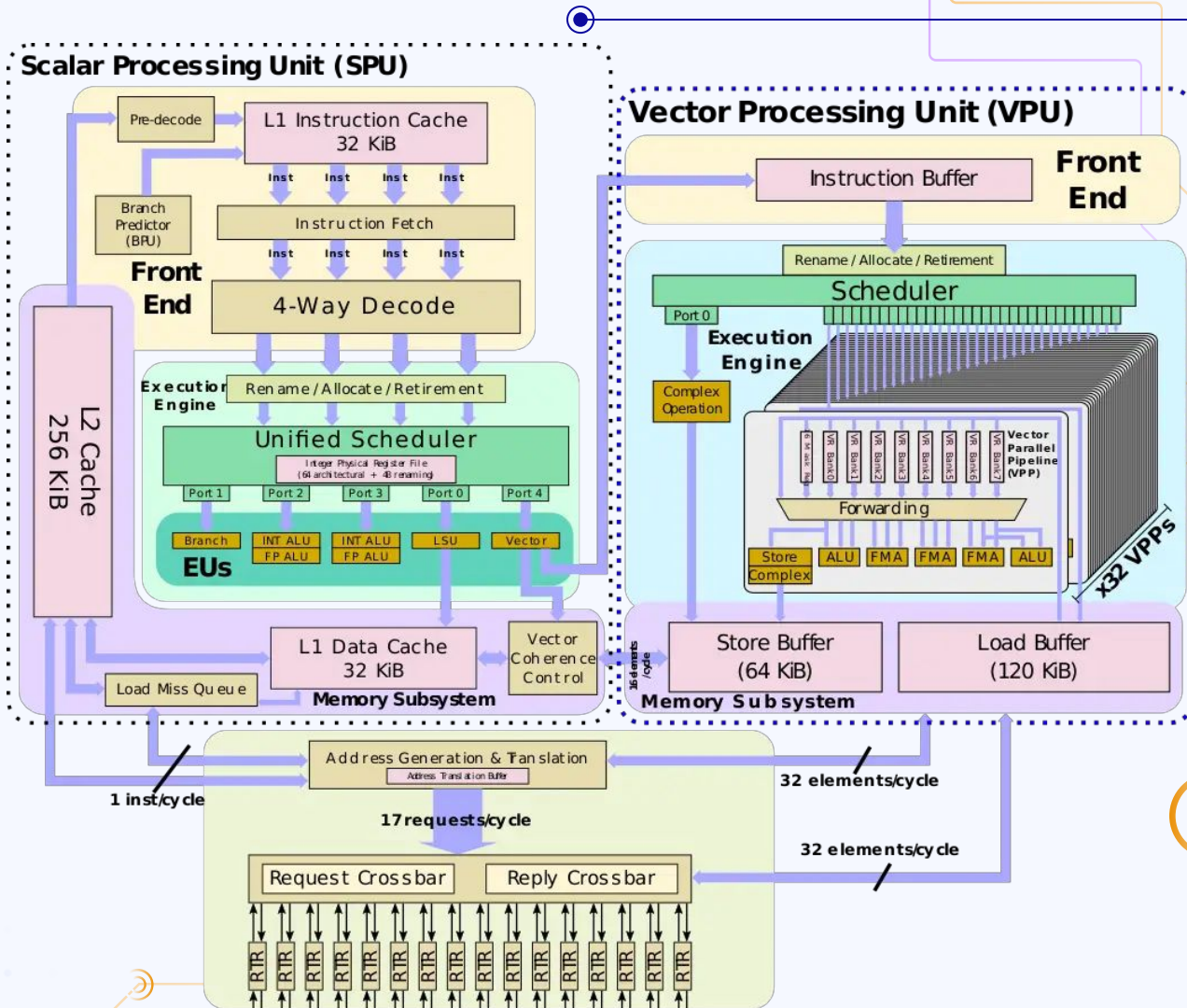


NUMA: non-uniform memory access
eaches only cache their local memory
writes to remote memory invalidate
corresponding ecache data

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

New York DFS to acquire supercomputer to understand and regulate AI

Also looking to hire professionals to run

July 05, 2023 By:



source

New York's Dep
dedicated to run

THE IRS IS BUYING AN AI SUPERCOMPUTER FROM NVIDIA

How exactly the IRS will use the Super Pod AI hardware is unclear. But it could be used for a variety of purposes, including tax enforcement.



source

OK fine.. but what about DLP for the rest of us?

NOAA completes upgrade to weather and climate supercomputer system

...nt upgrade to the 'American' forecast model

Share: [Twitter](#) [Facebook](#) [Email](#) [Print](#)

source

(also headlines about India, Japan, China, Germany, Brazil, ... in the past year)



...supercomputing system - just received a 20% upgrade.

Located in Mississippi, Virginia, and Tennessee, respectively, each supercomputer now operates at a speed of 14.5 petaflops. (Image credit: General Dynamics Information Technology (GDIT))

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

(Doesn't actually exist, but the letter "P" is reserved for such a thing)

Reuses floating-point registers

Packs multiple values in one register based on configuration

Ex: 64-bit register can hold 8 8-bit values, 4 16-bit values, 2 32-bit values, or 1 64-bit value

Requires special load/store operations

Hardware support for parallel operation on each value in register

ARMv6 SIMD

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

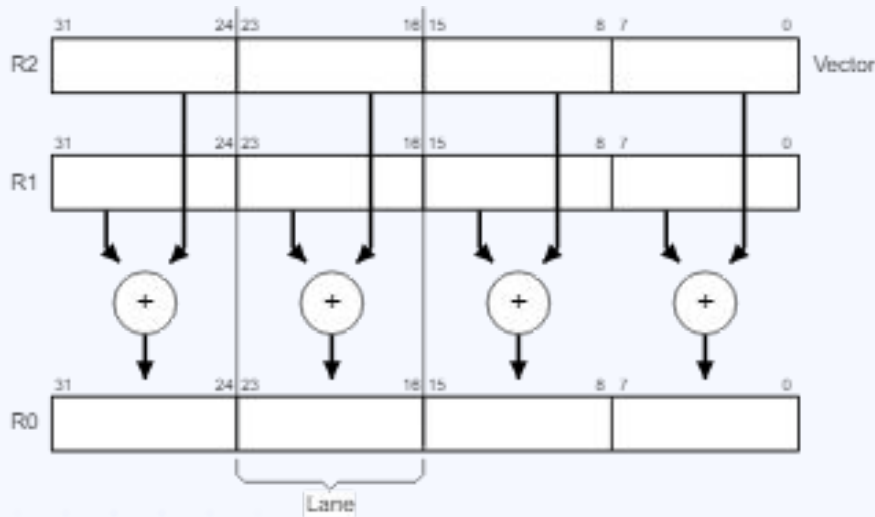


image source

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

Use of ARM NEON

Compilers are sometimes hit-or-miss when figuring out if they can vectorize code

Multimedia applications: people can use libraries

To get more flexibility than a library, ARM provides intrinsics

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)

Input image

9	4	1	2	2
1	1	1	0	4
1	2	1	0	6
1	0	0	2	1
9	6	7	4	1

Filter

0	2	1
4	1	0
1	0	1

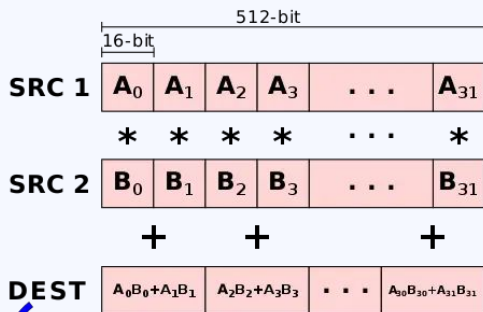
Output array

$$\begin{aligned} \text{Output}[0][0] &= (9*0) + (4*2) + (1*4) \\ &+ (1*1) + (1*0) + (1*1) + (2*0) + (1*1) \\ &= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1 \\ &= 16 \end{aligned}$$

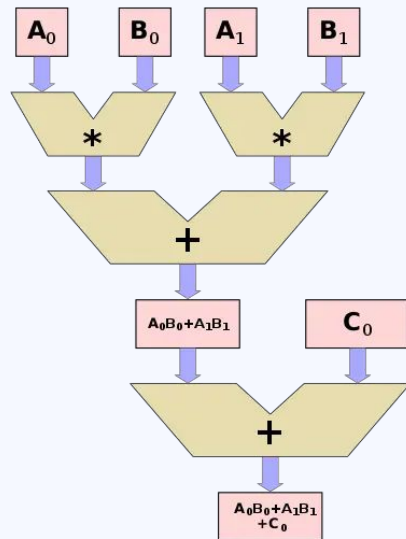
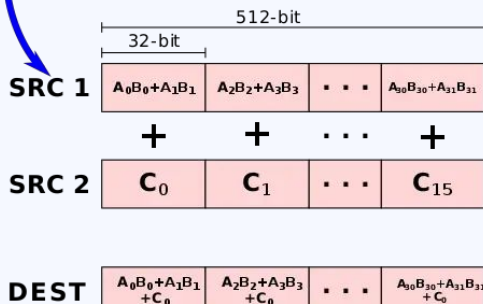
Image source

Image source

VPMADDWD



VPADDQ



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

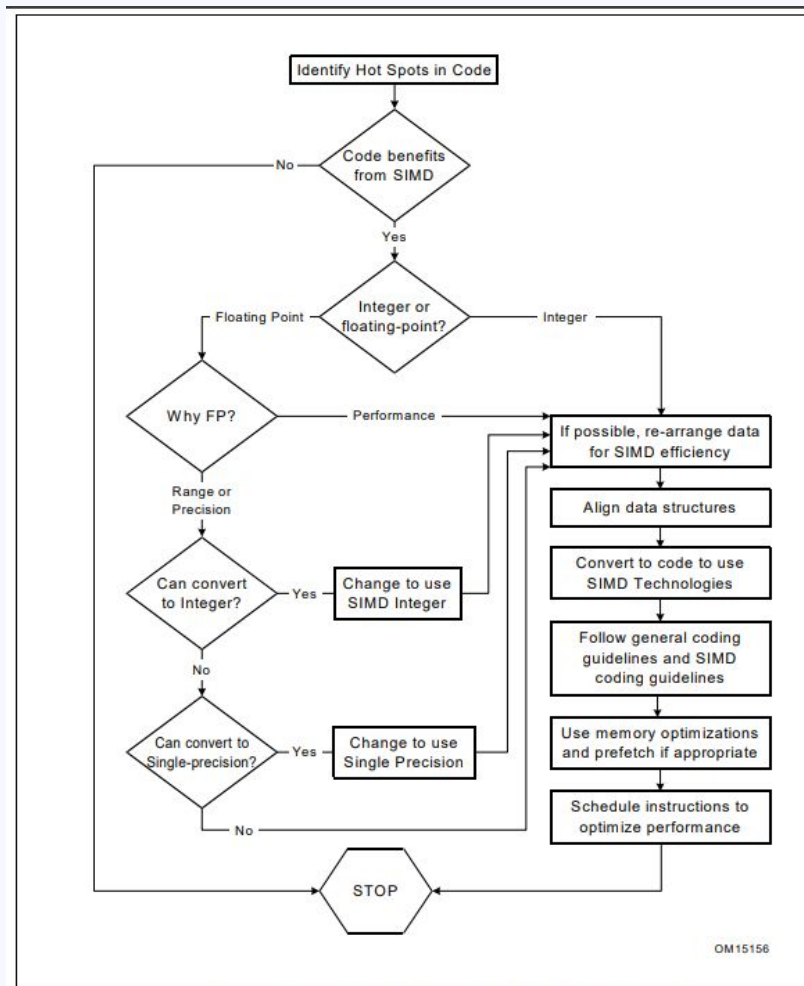


Figure 5-3. Converting to Intel® Streaming SIMD Extensions Chart