SIMD Instruction Set Extensions for Multimedia

- SIMD Multimedia Extensions started with the simple observation that many media applications operate on narrower data types than the 32-bit processors were optimized for. Many graphics systems used 8 bits to represent each of the three primary colors plus 8 bits for transparency. Depending on the application, audio samples are usually represented with 8 or 16 bits.
- By partitioning the carry chains within, say, a 256-bit adder, a processor could perform simultaneous operations on short vectors of thirty-two 8-bit operands, sixteen 16-bit operands, eight 32-bit operands, or four 64-bit operands.
- Like vector instructions, a SIMD instruction specifies the same operation on vectors of data. Unlike vector machines with large register files such as the VMIPS vector register, which can hold as many as sixty-four 64-bit elements in each of 8 vector registers, SIMD instructions tend to specify fewer operands and hence use much smaller register files.

- In contrast to vector architectures, which offer an elegant instruction set that is intended to be the target of a vectorizing compiler, SIMD extensions have three major omissions:

Instruction category	Operands
Unsigned add/subtract	**Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit**
Maximum/minimum	**Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit**
Average	**Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit**
Shift right/left	**Thirty-two 8-bit, sixteen 16-bit, eight 32-bit, or four 64-bit**
Floating point	**Sixteen 16-bit, eight 32-bit, four 64-bit, or two 128-bit**

Fig Summary of typical SIMD multimedia support for 256-bit-wide operations

- Multimedia SIMD extensions fix the number of data operands in the opcode, which has led to the addition of hundreds of instructions in the MMX, SSE, and AVX extensions of the x86 architecture. Vector architectures have a vector length register that specifies the number of operands for the current operation. These variable-length vector registers easily accommodate programs that naturally have shorter vectors than the maximum size the architecture supports.

- Multimedia SIMD does not offer the more sophisticated addressing modes of vector architectures, namely strided accesses and gather-scatter accesses. These features increase the number of programs that a vector compiler can successfully vectorize.
- Multimedia SIMD usually does not offer the mask registers to support conditional execution of elements as in vector architectures.
- For the x86 architecture, the MMX instructions added in 1996 repurposed the 64-bit floating-point registers, so the basic instructions could perform eight 8-bit operations or four 16-bit operations simultaneously. These were joined by parallel MAX and MIN operations, a wide variety of masking and conditional instructions, operations typically found in digital signal processors, and ad hoc instructions that were believed to be useful in important media libraries.
- The Streaming SIMD Extensions (SSE) successor in 1999 added separate registers that were 128 bits wide, so now instructions could simultaneously perform sixteen 8-bit operations, eight 16-bit operations, or four 32-bit operations. It also performed parallel single-precision floating-point arithmetic. Since SSE had separate registers, it needed separate data transfer instructions.
- The Advanced Vector Extensions (AVX), added in 2010, doubles the width of the registers again to 256 bits and thereby offers instructions that double the number of operations on all narrower data types

AVX Instruction	**Description**
VADDPD	Add four packed double-precision operands
VSUBPD	Subtract four packed double-precision operands
VMULPD	Multiply four packed double-precision operands
VDIVPD	Divide four packed double-precision operands
VFMADDPD	Multiply and add four packed double-precision operands
VFMSUBPD	Multiply and subtract four packed double-precision operands
VCMPxx	Compare four packed double-precision operands for EQ, NEQ, LT, LE, GT, GE,

VMOVAPD	Move aligned four packed double-precision operands
VBROADCASTSD	Broadcast one double-precision operand to four locations in a 256-bit register

Fig: AVX instructions for x86 architecture useful in double-precision floating-point programs.

The Roofline Visual Performance Model :

- One visual, intuitive way to compare potential floating-point performance of variations of SIMD architectures is the Roofline model [Williams et al. 20091. It ties together floating-point performance, memory performance, and arithmetic intensity in a two-dimensional graph. Arithmetic intensity is the ratio of floating-point operations per byte of memory accessed.
- It can be calculated by taking the total number of floating-point operations for a program divided by the total number of data bytes transferred to main memory during program execution.
- Peak floating-point performance can be found using the hardware specifications. Many of the kernels in this case study do not fit in on-chip caches, so peak memory performance is defined by the memory system behind the caches.

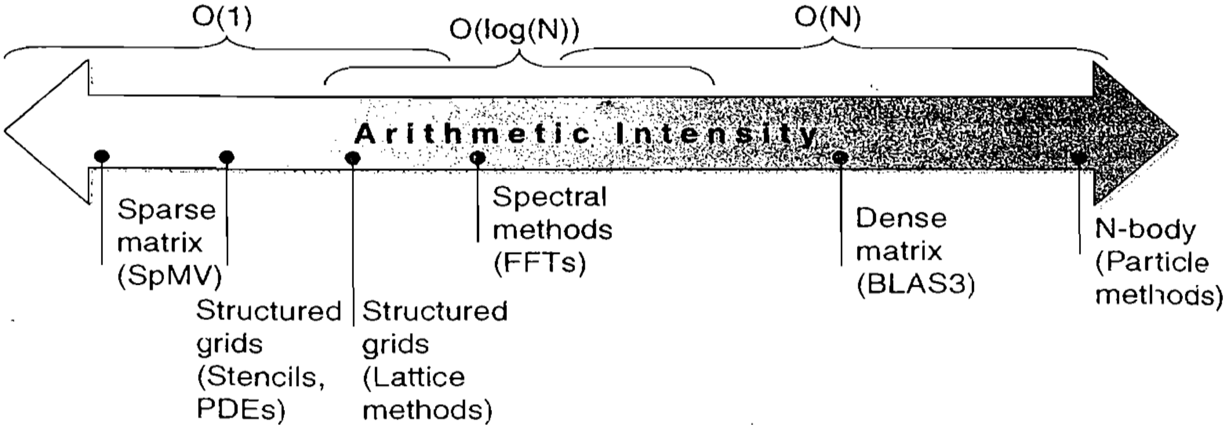

Fig : Arithmetic intensity

- Fig below shows the Roofline model for the NEC SX-9 vector processor on the left and the Intel Core i7 920 multicore computer on the right. The vertical Y-axis is achievable floating-point performance from 2 to 256 GFLOP/sec. The horizontal X-axis is arithmetic intensity, varying from 1/8th FLOP/DRAM byte accessed to 16 FLOP/ DRAM byte accessed in both graphs. Note that the graph is a log-log scale, and that Rooflines are done just once for a computer.

- For a given kernel, we can find a point on the X-axis based on its arithmetic intensity. If we drew a vertical line through that point, the performance of the kernel on that computer must lie somewhere along that line. We can plot a horizontal line showing peak floating-point performance of the computer. Obviously, the actual floating-point performance can be no higher than the horizontal line, since that is a hardware limit.

- How could we plot the peak memory performance? Since the X-axis is FLOP/ byte and the Y-axis is FLOP/sec, bytes/sec is just a diagonal line at a 45-degree angle in this figure. Hence, we can plot a third line that gives the maximum floating-point performance that the memory system of that computer can support for a given arithmetic intensity. We can express the limits as a formula to plot these lines in the graphs.

Attainable GFLOPs/sec = Min(Peak Memory BW x Arithmetic Intensity, Peak Floating-Point Perf.)

- The horizontal and diagonal lines give this simple model its name and indicate its value. The "Roofline" sets an upper bound on performance of a kernel depending on its arithmetic intensity. If we think of arithmetic intensity as a pole that hits the roof, either it hits the flat part of the roof, which means performance is computationally limited, or it hits the slanted part of the roof, which means performance is ultimately limited by memory bandwidth.

- The vertical dashed line on the right (arithmetic intensity of 4) is an example of the former and the vertical dashed line on the left (arithmetic intensity of 1/4) is an example of the latter. Given a Roofline model of a computer, you can apply it repeatedly, since it doesn't vary by kernel.

- The "ridge point," where the diagonal and horizontal roofs meet, offers an interesting insight into the computer. If it is far to the right, then only kernels with very high arithmetic intensity can achieve the maximum performance of that computer. If it is far to the left, then almost any kernel can potentially hit the maximum performance.

- The peak computational performance of the SX-9 is 2.4x faster than Core i7, but the memory performance is l Ox faster. For programs with an arithmetic intensity of 0.25, the SX-9 is l0x faster (40.5 versus 4.1 GFLOP/sec). The higher memory bandwidth moves the ridge point from 2.6 in the Core i7 to 0.6 on the SX-9, which means many more programs can reach peak computational performance on the vector processor

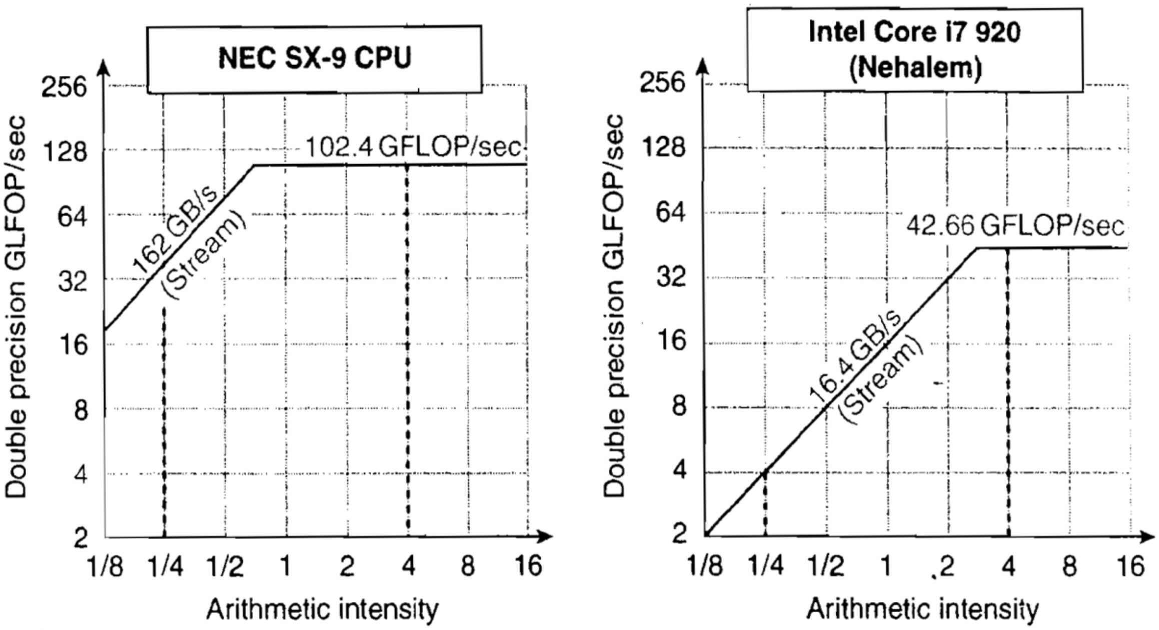

Fig: **Roofline model for one NEC SX-9 vector processor on the left and the Intel Core i7 920 multicore computer with SIMD Extensions on the right.**