

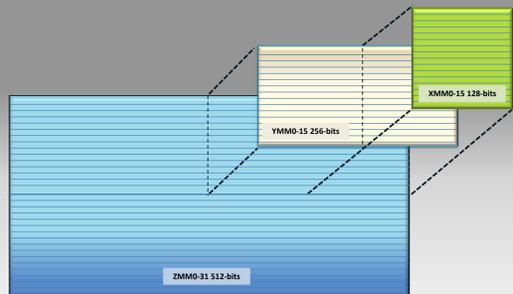


# Intel® AVX-512 Architecture

Comprehensive vector extension for HPC and enterprise

Elena Demikhovskiy  
Intel® Software and Services Group  
Israel

## More And Bigger Registers

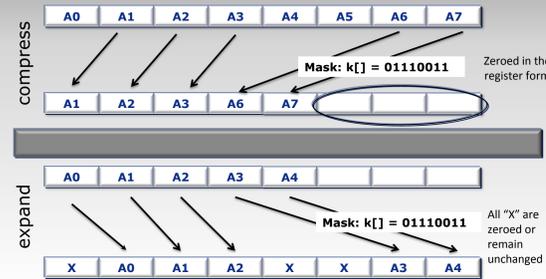


32 SIMD registers 512 bit wide

## Compress And Expand

```
Compress values from 512-bit vectors compound of f64, f32, i64, i32
elements using mask and store in register or memory
VCOMPRESSDPS zmm1/mV {k1}, zmm2
VEXPANDPS zmm1 {k1}{z}, zmm2/mV
```

```
for (i=0; i < N; i++) {
  if (topVal > b[i]) {
    *dst = a[i];
    dst++;
  }
}
```



## Conflict Detection

Sparse computations are hard for vectorization

```
for(i=0; i<16; i++) { A[B[i]]++; }

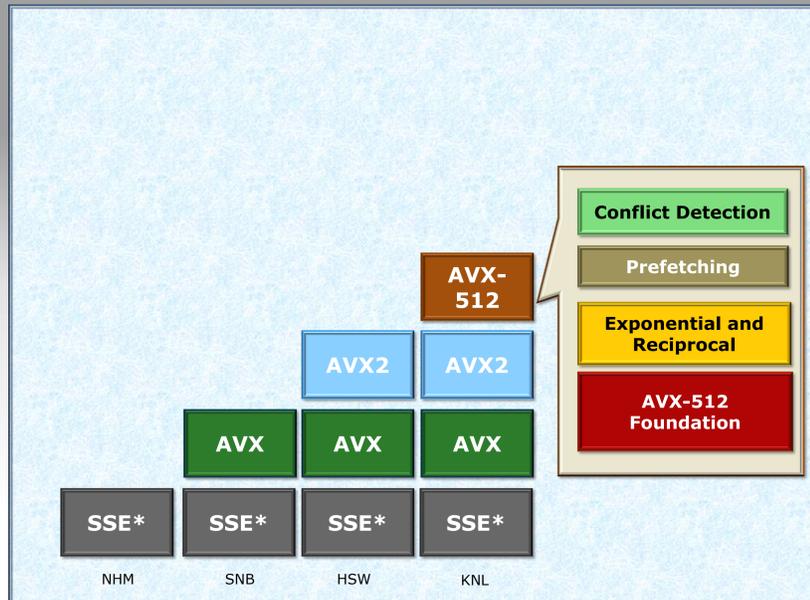
index = vload &B[i] // Load 16 B[i]
old_val = vgather A, index // Grab A[B[i]]
new_val = vadd old_val, +1.0 // Compute new values
vscatter A, index, new_val // Update A[B[i]]
```

Code above is wrong if any values within B[i] are duplicated

VPCONFLICT instruction detects elements with conflicts

```
index = vload &B[i] // Load 16 B[i]
pending_elem = 0xFFFF;
do {
  curr_elem = get_conflict_free_subset(index, pending_elem)
  old_val = vgather {curr_elem} A, index // Grab A[B[i]]
  new_val = vadd old_val, +1.0 // Compute new values
  vscatter A {curr_elem}, index, new_val // Update A[B[i]]
  pending_elem = pending_elem ^ curr_elem // remove done idx
} while (pending_elem)
```

## AVX-512 – What's new?



- 512-bit wide vectors, 32 SIMD registers
- 8 new mask registers
- Embedded Rounding Control
- Embedded Broadcast
- New Math instructions
- 2-source shuffles
- Gather and Scatter
- Compress and Expand
- Conflict Detection

## Embedded Broadcast

A source from memory is repeated across all the elements.

```
vbroadcastss zmm3, [rax]
vaddps zmm1, zmm2, zmm3
↓
vaddps zmm1, zmm2, [rax] {1to16}
```

## Embedded Rounding Control

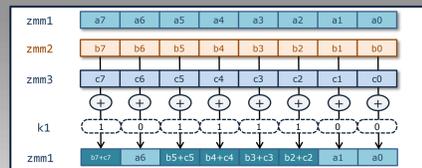
- Static (per instruction) rounding mode
- No need to access MXCSR any more!

```
vaddps zmm7 {k6}, zmm2, zmm4 {rd}
vcvtdq2ps zmm1, zmm2, {ru}
```

All exceptions are always suspended by using embedded RC

## Masking

Unmasked elements remain unchanged:  
VADDPD zmm1 {k1}, zmm2, zmm3  
Or zeroed:  
VADDPD zmm1 {k1} {z}, zmm2, zmm3



- Memory fault suppression
- Avoid FP exceptions
- Avoid extra blends

```
float32 A[N], B[N], C[N];
for (i=0; i<16; i++) {
  if (B[i] != 0)
    A[i] = A[i] / B[i];
  else
    A[i] = A[i] / C[i];
}
```

```
VMOVUPS zmm2, A
VCMPPS k1, zmm0, B
VDIVPS zmm1 {k1}{z}, zmm2, B
KNOT k2, k1
VDIVPS zmm1 {k2}, zmm2, C
VMOVUPS A, zmm1
```

## Masking in LLVM

### Predication Scheme

```
Source code
If (condition) {
  A = ..
} else {
  A = ..
}

LLVM IR
%Mask = cmp (condition)
...
%A1 =
...
%A2 =
%A = SELECT %Mask, %A1, %A2
```

```
Machine code
CMP %regMask, ops
...
ADD %regA1, x1, y1
...
ADD %regA2 = x2, y2
BLEND %regA, %regMask, %regA1, %regA2
```

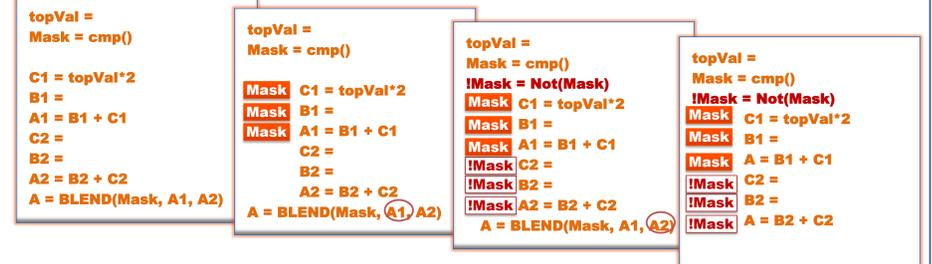
Goal: To set predicates for instructions that calculate A1 and A2 (Mask and Not-Mask)

Result: If the mask is all-zero, instruction will not be executed.

```
Mask = cmp (condition)
Not-Mask = not (Mask)
{Mask} C1 =
{Mask} B1 =
{Mask} A = B1+C1

{Not-Mask} C2 =
{Not-Mask} B2 =
{Not-Mask} A = B2+C2
```

### Mask Propagation Pass - design ideas



A new Machine Pass:

- Before register allocation
- Start from the "blend" operands and go up recursively till mask definition
- Check all users of the destination operand before applying the mask

Mask Propagation Pass does not guarantee full mask propagation over the whole path from blend to compare

- Load/Store operations require exact masking
- FP operations require masking if exceptions are not suppressed
  - IR generators should use compiler intrinsics

Predicated memory accesses in LLVM IR would be helpful!



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS. Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information. The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order. Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm. Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Copyright © 2013 Intel Corporation. All rights reserved. Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.