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

Conflict Detection

Sparse computations are hard for vectorization

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

index = vload 4B[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

```cpp
index = vload 4B[i] // Load 16 B[i]
pending_alm = 0x0; do { 
curr_alm = get_conflict_free_subset(index, pending_alm); old_val = vgather A[curr_alm], index // Grab A[B(i)]
new_val = vadd old_val, -1.0 // Compute new values
vscatter A[curr_alm], index, new_val // Update A[B[i]]
pending_alm = pending_alm + curr_alm // remove done idx }
```

Masking in LLVM

Unmasked elements remain unchanged:
VADDPPD zmm1 (k1), zmm2, zmm3

Or zeroded:
VADDPPD zmm1 (k1) [z], zmm2, zmm3

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

Predication Scheme

<table>
<thead>
<tr>
<th>Source Scheme</th>
<th>LLVM IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>in, i</td>
<td>zmm</td>
</tr>
<tr>
<td>A = cmp (condition)</td>
<td>zmm</td>
</tr>
<tr>
<td>A = ...</td>
<td></td>
</tr>
</tbody>
</table>

Machine code

```cpp
CMP long long, npe
ADD long long, npe
```

Smask = cmp (condition)

```cpp
M = Select (Smask, A, B) |
```

A new Mask Propagation Pass
- Before register allocation
- Start from the 'blend' operands and go up recursively till mask definition
- Check all users of the destruction 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!