Intel® AVX-512 architecture evolution and support in Clang/LLVM

Available in trunk since July 2014!
clang -march=kel... clang -march=skx...

New features

AVX-512VL: Vector Length Orthogonality

- Apply AVX-512F instructions to 128b (%XMM) and 256b (%YMM) registers
- AVX-512F (starting with Xeon Phi)
  - VADDPD (%YMM2, %YMM2, %YMM3)
  - AVX-512F(VL) (starting with Skylake Xeon)
  - VADDPD (%YMM2, %YMM2, %YMM3)
  - VADDPD (%YMM2, %YMM2, %YMM3)

AVX-512BW: Byte & word support

- AVX-512F packed instructions work on double and quad words

AVX-512DQ: New HPC Instructions

- Extended Tuple support: 32x8, 64x2, 32x2
- Intel® FP conversions
- Transcendental package enhancements
- INT64 arithmetic support
- Byte support for mask instructions
- Expanded mask functionality

AVX-512 in Clang/LLVM

- Total: 651 instructions, 4000+ intrinsics
- 30% of these instructions implemented
- Encodings, lowering and intrinsics covered with tests
- 100+ patches, 9000+ LOCs
- Work in progress!

Enabling compiler optimizations

- With AVX-512 we can generate this smart code!

VCHMPNEQDPS k1, zmm0, B
VMOVUPS zmm1[ k1], A
VMOVPAPS A[k1], zmm1

Currently, this loop can't be vectorized in LLVM IR:

for(i=0; i<16; i++) {
  // Peeled part
}

Potential LLVM IR extended with special intrinsics for masking

%a.call <16 x float>* @llvm.masked.load %a.ptr, <16 x int1> %mask, <16 x float>* @llvm.masked.store %a.ptr, <16 x float>* %a.ptr, <16 x int1> %mask

Vectorization of Peeled Loops

- float A[N], B[N], C[N];
- if(N<0; i<16; i++) {
  if(B[i] == 0) {
    C[i] = A[i] * B[i];
  }
}

More samples

Elena Demikhovsky's Intel® AVX-512 Architecture review poster @ 2013 LLVM DevMtg

Kril Yuksin's Intel® Advanced Vector Extensions 2015/16 Support in GNU Compiler Collection @ GNU Tools Cauldron 2014