Supporting TVM on RISC-V Architectures

Jenq-Kuen Lee[1] , Allen Lu[2] , Yuan-Ming Chang[1,2], Chao-Lin Lee[1,2]
Piyo Chen[1], and Shao-Chung Wang[3]

[1]Department of Computer Science, National Tsing Hua University, Taiwan
[2]Peakhills Group Corporation
[3]Andes Technology Corporation

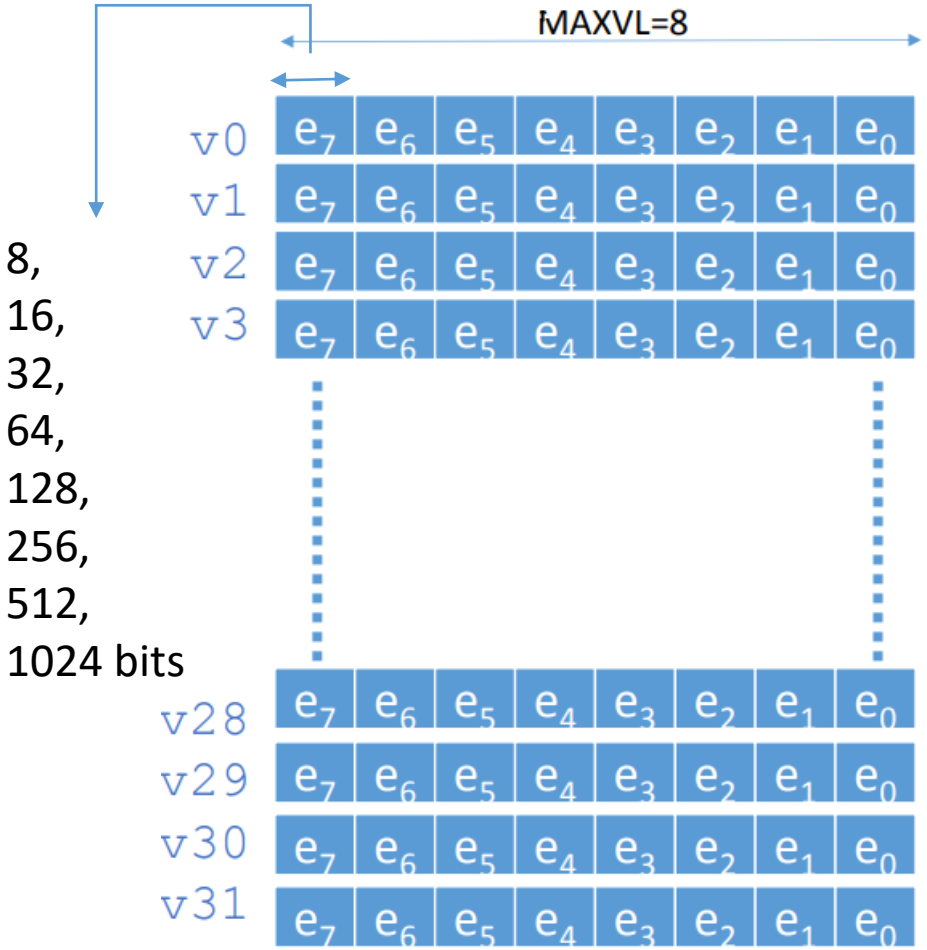
PEAKHILLSGROUP

ANDES TECHNOLOGY

NTHU PL LAB

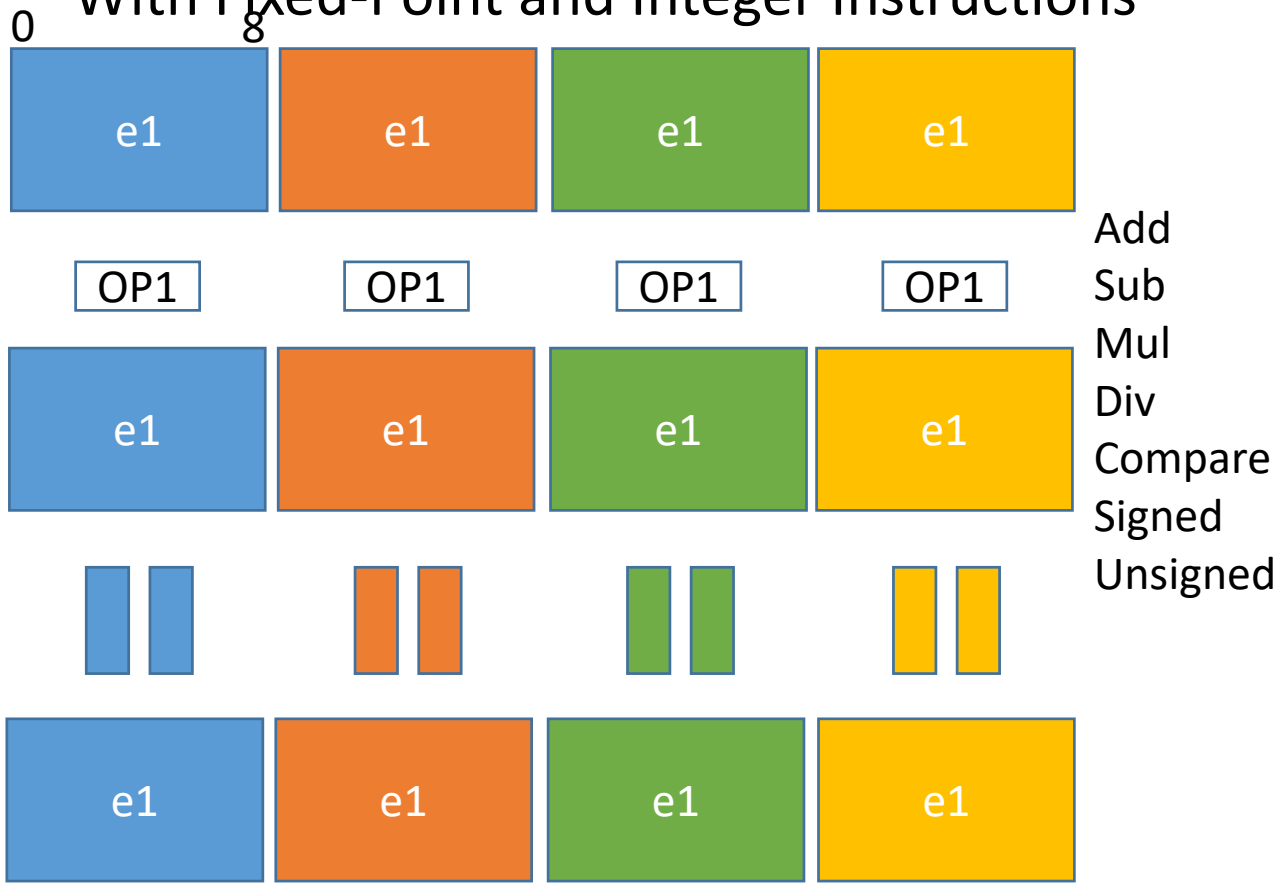
RISC-V with two vector ISAs to support fall-back engine with AI Models

Super Word Vector



Courtesy: Vector ISA, Roger Espasa, Esperanto Technologies

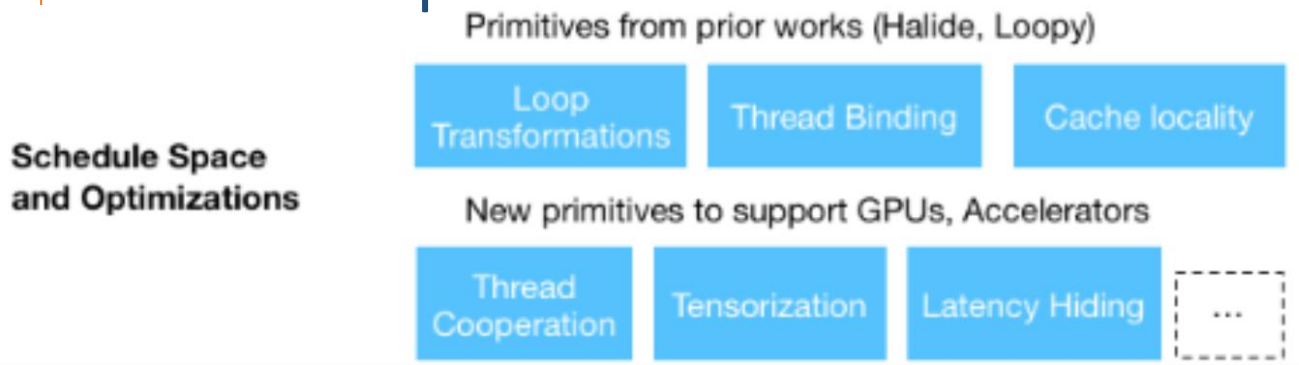
Packed Vector (SubWord SIMD) With Fixed-Point and Integer Instructions



RISC-V DSP (P) Extension Proposal Chuan-Hua Chang, Andes Technology Corporation

Support TVM on RISC-V with Subword SIMD Computation

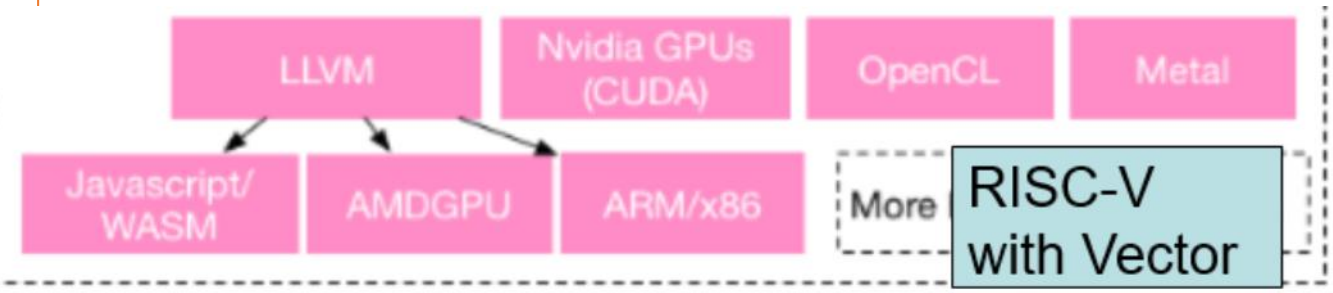
- We add RISC-V target in TVM codegen phase. The TVM RISC-V codegen will lower SIMD computation with Subword SIMD intrinsics.
- The LLVM backend will need to generate the corresponding SIMD instructions.
- Also on-going work to add TVM scheduling to quantize computation into fixed-points, “quantize(width, exponent)”.



New Primitives support RISC-V with vector units

Subword SIMD Quantization

SIMD Rewriting with Intrisnsics



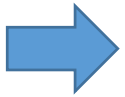

```
Expr CodeGenRISCV::CreateAddIntr(const Add* op,

    ::llvm::Intrinsic::ID vadd_id = ::llvm::Int
    if(bits==8) {
      vadd_id=::llvm::Intrinsic::riscv_simd_sad
    }
    const Expr& e1 = op->a;
    const Expr& e2 = op->b;
    Array<Expr> vcnt_args;
    vcnt_args.push_back(ir::UIntImm::make(UInt(32), vpaddlu_id));
    vcnt_args.push_back(ir::UIntImm::make(UInt(32), 0));
    vcnt_args.push_back(e1);
    vcnt_args.push_back(e2);
    return ir::Call::make(op->type,  "llvm_intrin", vcnt_args, Call::PureIntrinsic);
}
```

tvm/src/codegen/llvm/codegen_riscv.cc

Example – Matrix Multiply

```
produce compute {
  for (i.outer, 0, ((n + 1)/2)) {
    for (j, 0, n) {
      for (i.inner.s, 0, 2) {
        if (likely(((i.outer*2) < (n - i.inner.s)))) {
          compute[((((i.outer*2) + i.inner.s)*n) + j)]
          = (int16)0
        }
      }
      for (k, 0, n) {
        for (i.inner.s, 0, 2) {
          if (likely(((i.outer*2) < (n - i.inner.s)))) {
            compute[((((i.outer*2) + i.inner.s)*n) + j)]
            = (compute[((((i.outer*2) + i.inner.s)*n) + j)]
            + (A[((((i.outer*2) + i.inner.s)*n) + k)]*B[((j*n) + k)]))
          }
        }
      }
    }
  }
}
```

```
%22 = bitcast i8* %21 to i16*
%23 = load i16, i16* %4, align 2, !tbaa !115
%24 = insertelement <2 x i16> undef, i16 %23, i32 0
%25 = shufflevector <2 x i16> %24, <2 x i16> undef,
      <2 x i32> zeroinitializer
%26 = tail call <2 x i16> @llvm.riscv.simd.khm16
      (<2 x i16> %13, <2 x i16> %25)
%27 = tail call <2 x i16> @llvm.riscv.simd.sadd16
      (<2 x i16> zeroinitializer, <2 x i16> %26)
```



```
pkbb16  a6, a7, a6
lhu a7, 0(a2)
li  a5, 7
li  a3, 8
pkbb16  a4, a4, a4
ksll16  a4, a4, a3
ksll16  a6, a6, a5
khm16 t2, a6, a4
pkbb16  a4, a7, a7
pkbb16  a7, t1, t0
lhu t1, 16(a2)
ksll16  a4, a4, a3
ksll16  a7, a7, a5
lhu t3, 4(a1)
lhu t4, 12(a1)
khm16 t0, a7, a4
pkbb16  a4, zero, zero
add16 t0, a4, t0
```

In this example, **104 of 229** instructions will be with SIMD computation which process **two element** in one instruction.

Subword SIMD Intrinsic LLVM IR

Summary and Future Work

- Also has some discussions with AWS team to add RISC-V back-end for TVM deep learning compiler.
- Look forward to contributing the codes to TVM source trees.
- Currently the work is with Spike RISC-V simulator and we look forward to using Gem5 and Sid simulators and real chips for performance tuning.

PEAKHILLSGROUP

ANDES TECHNOLOGY

NTHU PL LAB