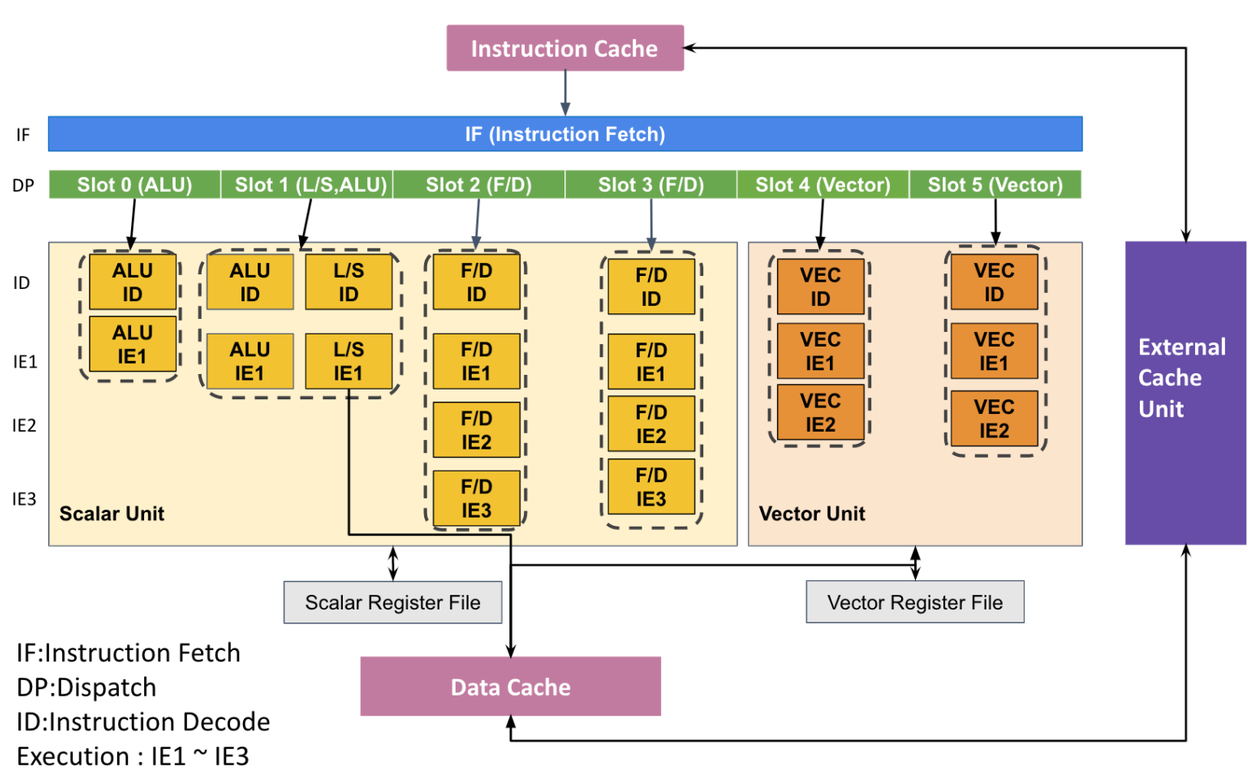
Developing An LLVM Backend For VLIW RISC-V Vector Extention Architectures

Hao-Chun Chang, Meng-Shiun Yu, Tai-Liang Chen,
Jhih-Kuan Lin, Jenq-Kuen Lee
Department of Computer Science
National Tsing-Hua University, Taiwan

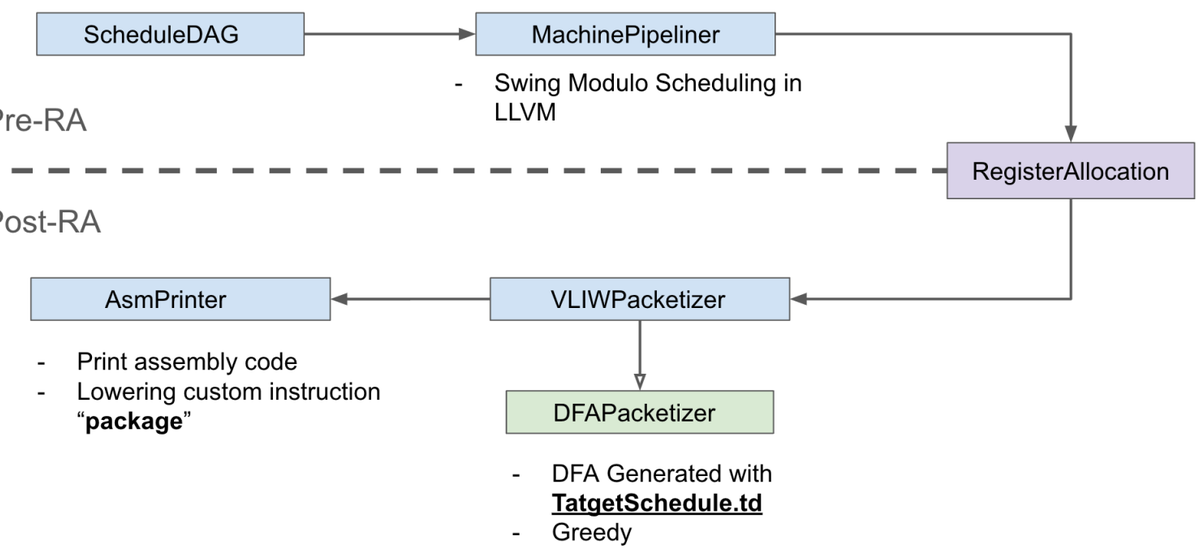
VLIW RISC-V Vector Extension Architecture



IF:Instruction Fetch
DP:Dispatch
ID:Instruction Decode
Execution : IE1 ~ IE3

LLVM Backend

a. Codegen flow



In this code generation flow, we add DFAPacketizer to group instructions into an instruction bundle and rewrite RISCVAsmPrinter to emit instruction bundle with the custom instruction we use to imply the begining of the bundle.

b. Codegen example

```
for(int i = 0 ; i < X ; i++)
{
   s = A[i];
   A[i] = s * 42 + 3;
}
```

```
        package zero, 0
        li      a3, 32
        package zero, 0
        bgeu    a1, a3, .LBB1_11
# %bb.10:
        package zero, 0          <- Custom Instruction we add to imply instruction bundle
        li      a1, 0
        j       .LBB1_15
.LBB1_11:
        package zero, 0          <- Instruction bundle
        mv      a1, a0
        addi    a4, sp, 16
        package zero, 0
        li      a5, 128
        vsetvli a6, zero, e32, m8, ta, mu
        vmv.v.i v8, 3
        package zero, 0
        li      a6, 42
        mv      a7, a0
.LBB1_12:

        package zero, 0
        vl8re32.v       v16, (a4)
        package zero, 0
        vmadd.vx        v16, a6, v8
        package zero, 0          <- Instruction bundle
        vs8r.v  v16, (a4)
        sub     a7, a7, a3
        add     a4, a4, a5
        package zero, 0
        bnez    a7, .LBB1_12
```

ResMII with LMUL Factor for Swing Modulo Scheduling

	iteration 1	iteration 2	iteration 3	iteration 4	iteration 5
cycle1	LD				
cycle2	VADD	LD			
cycle3	VMUL	VADD	LD		
cycle4	ST	VMUL	VADD	LD	
cycle5		ST	VMUL	VADD	LD
cycle6			ST	VMUL	VADD
cycle7				ST	VMUL
cycle8					ST
cycle9					

Assume each instruction latency is 1, the ResMII will be set to 1. However, the LMUL will group multiple registers and affect operation latency in some processor implementation.
The resource conflict might happened at cycle 3 when the latency of vector operation is larger than 1. As a result, the ResMII should be modified.

Algorithm 1: The algorithm to evaluate ResMII with consideration of LMUL factor

Input : The maximum register pressure: $MaxPressure$
number of spill register: S
Total Latnecy of vector unit instruction: LV
Total Latency of other unit: LO
parameter: The number of registers in the register file: NR=32
Constant value : c
LMUL factor that affects latency: $LMUL\_Fac$
Number of vector units: $VectorUnits$
Number of other function units: $OtherUnits$
Output : Feasible LMUL value: $LMUL$
ResourceMII: $ResMII$

begin
  $LMUL \leftarrow \frac{NR}{MaxPressure} \times \frac{c}{2^S}$
  if $LMUL \leq 1$ **then**
    $LMUL \leftarrow 1$
  end
  $ResMII \leftarrow \max\{\frac{LV*LMUL\_Fac}{VectorUnits}, \frac{LO}{OtherUnits}\}$
end

In this algorithm of deciding the feasible ResMII, we calculate the feasible LMUL first, and it is done by dividing the number of vector registers with the maximum register pressure and square of number of spill registers. Then the LMUL will be considered as a factor to get the feasible ResMII.

Experiment Results

Benchmark : DSPstone
LLVM version : LLVM 15
Simulator : Spike