# RVV 0.7.1与0.9之间的差异

## 配置指令

1. vsetvli支持的汇编常量有增加

0.7.1中有e8/16/32/64/128，m1/2/4/8，d1/2/4/8

0.9中有e8/16/32/64/128/256/512/1024，m1/2/4/8/f2/f4/f8，ta/tu，ma/mu

## 加载/存储指令

1. 第7章节（7. Vector Loads and Stores）加载/存储指令在0.9版本被重新设计，包括指令名、指令格式和编码等。

## Zvamo扩展指令

1. 第8章节（8. Vector AMO Operations (Zvamo)）原子内存操作指令在0.9版本被重新设计，包括指令名和编码等。
2. AMO指令的操作数顺序有调整。在0.7.1版本中，指令形式如下：

vamoaddw.v v4, v8, (a1), v4

在0.9版本中，指令形式如下：

vamoaddei32.v x0, (rs1), v2, vs3, v0.t

## 整数算术指令

1. 在0.9版本中，增加了整数扩展指令（12.3. Vector Integer Extension）

vzext.{vf2, vf4, vf8}

vsext.{vf2, vf4, vf8}

1. 在0.9版本中，进位加/借位减（12.4. Vector Integer Add-with-Carry / Subtract-with- Borrow Instructions），对于vmadc和vmsbc指令，增加了后缀为vv/vx的情况：

vmadc.{vv, vx}

vmsbc.{vv, vx}

1. 对于收缩右移指令（Vector Narrowing Integer Right Shift Instructions），

在0.7.1版本中（12.6章节），指令后缀为：

vnsrl.{vv, vx, vi}

vnsra.{vv, vx, vi}

在0.9版本中（0.9中的12.7章节），指令后缀为：

vnsrl.{wv, wx, wi}

vnsra.{wv, wx, wi}

## 定点算术指令

1. 在0.7.1版本中，单宽度平均加减指令（13.2. Vector Single-Width Averaging Add and Subtract），对于vaadd，有vi后缀指令。在0.9版本中被去掉了。

vaadd.vi

1. 在0.9版本中，单宽度平均加减指令（13.2. Vector Single-Width Averaging Add and Subtract），增加了无符号运算：

vaaddu.{vv, vx}

vaadd.{vv, vx}

vasubu.{vv, vx}

vasub.{vv, vx}

1. 在0.7.1版本中，有扩展饱和乘加指令（13.4. Vector Widening Saturating Scaled Multiply-Add），在0.9版本中被删减。

vwsmaccu.{vv, vx}

vwsmacc.{vv, vx}

vwsmaccsu.{vv, vx}

vwsmaccus.{vx}

相关git log：

commit 063b128bd91390c64796fe1e1546a8855fdbaf35

Author: Krste Asanovic <krste@eecs.berkeley.edu>

Date: Fri Nov 15 09:49:13 2019 -0800

Removed scaled fixed-point multiply-add instructions.

Task group agreed that fixed scale amounts were of limited utility.

1. 对于收缩定点Clip指令（Vector Narrowing Fixed-Point Clip Instructions），在0.7.1版本中（13.6），指令后缀为：

vnclipu.{vv, vx, vi}

vnclip.{vv, vx, vi}

在0.9版本中（13.5），指令后缀为：

vnclipu.{wv, wx, wi}

vnclip.{wv, wx, wi}

## 浮点指令

1. 在0.7.1版本中，浮点比较指令（14.11. Vector Floating-Point Compare Instructions）提供了vmford指令。

vmford.{vv, vf}

在0.9版本中，该指令被删减。

相关git log：

commit 13081557f1a1ecc4d632cec376ee98e29d46ff2b

Author: Andrew Waterman <andrew@sifive.com>

Date: Tue Jul 16 03:21:29 2019 -0700

Remove the vmford instruction (#249)

@jhauser-us's survey of the C math library functions concluded that vmford

will not have sufficient impact on performance to justify including. Many

of the quiet comparisons are against a constant, in which case it does not

help. For the remaining ones, the penalty is only 1-2 instructions,

depending on whether signaling NaNs are supported. While signaling NaNs

are supported within the C library itself, GCC's default is to not support

them, so compiled code incurs only a one-instruction penalty for lacking

the vmford instruction.

1. 对于浮点/整数类型转换指令（Single-Width/Widening/Narrowing Floating-Point/Integer Type-Convert Instructions），在0.9版本中，增加了rtz（round towards zero）和rod（round towards odd）指令，同样对于收缩指令后缀名使用w。

在0.7.1版本中，相应章节的指令为：

14.14. Single-Width Floating-Point/Integer Type-Convert Instructions

vfcvt.{xu.f.v, x.f.v, f.xu.v, f.x.v}

14.15. Widening Floating-Point/Integer Type-Convert Instructions

vfwcvt.{xu.f.v, x.f.v, f.xu.v, f.x.v, f.f.v}

14.16. Narrowing Floating-Point/Integer Type-Convert Instructions

vfncvt.{xu.f.v, x.f.v, f.xu.v, f.x.v, f.f.v}

在0.9版本中，相应章节的指令为：

14.15. Single-Width Floating-Point/Integer TypeConvert Instructions

vfcvt.{xu.f.v, x.f.v, rtz.xu.f.v, rtz.x.f.v, f.xu.v, f.x.v}

14.16. Widening Floating-Point/Integer Type-Convert Instructions

vfwcvt.{xu.f.v, x.f.v, rtz.xu.f.v, rtz.x.f.v, f.xu.v, f.x.v, f.f.v}

14.17. Narrowing Floating-Point/Integer Type-Convert Instructions

vfncvt.{xu.f.w, x.f.w, rtz.xu.f.w, rtz.x.f.w, f.xu.w, f.x.w, f.f.w, rod.f.f.w}

## 规约操作

无

## Mask指令

1. 在0.7.1版本中，mask population count指令和find-first-set mask bit指令的名字为：

vmpopc.m

vmfirst.m

在0.9版本中，指令名为：

vpopc.m

vfirst.m

## 置换指令

1. 在0.7.1版本中，有整数抽取指令（17.1. Integer Extract Instruction）：

vext.x.v

在0.9版本中被删减。

相关git log：

commit 83fc27897b7b1fbc68e2e9e94f2ee05766315bac

Author: Andrew Waterman <andrew@sifive.com>

Date: Sun Jul 14 16:11:31 2019 -0700

Replace vext.x.v with vmv.x.s

This regularizes the integer moves with the floating-point ones, and

simplifies the implementation.

The effect of vext.x.v can be obtained with vslidedown.vx followed by

vmv.x.s.

Closes #154

1. 在0.7.1版本中，整数标量移动指令（17.2. Integer Scalar Move Instruction）只有一种形式：

vmv.s.x

在0.9版本中，提供了两种形式：

vmv.{x.s, s.x}

1. 在0.7.1版本中，slide1up和slide1down指令只提供了一种形式：

vslide1up.vx

vslide1down.vx

在0.9版本中，提供了两种形式

vslide1up.vx

vfslide1up.vf

vslide1down.vx

vfslide1down.vf

1. 在0.9版本中，增加了整个向量寄存器移动指令（17.6. Whole Vector Register Move）：

vmv<nr>r.v

## 指令编码

Integer, OPIVV, OPIVX, OPIVI

|  |  |  |
| --- | --- | --- |
| funct6 | 0.7.1 | 0.9 |
| 100100 | vaadd | - |
| 100110 | vasub | - |
| 111100 | vwsmaccu | vqmaccu |
| 111101 | vwsmacc | vqmacc |
| 111110 | vwsmaccsu | vqmaccus |
| 111111 | vwsmaccus | vqmaccsu |

Integer, OPMVV, OPMVX

|  |  |  |
| --- | --- | --- |
| funct6 | 0.7.1 | 0.9 |
| 001000 | - | vaaddu |
| 001001 | - | vaadd |
| 001010 | - | vasubu |
| 001011 | - | vasub |
| 001100 | vext.x.v | - |
| 001101 | vmv.s.x | - |
| 010100 | vmpopc | VMUNARY0 |
| 010101 | vmfirst | - |
| 010110 | VMUNARY0 | - |

FP, OPFVV, OPFVF

|  |  |  |
| --- | --- | --- |
| funct6 | 0.7.1 | 0.9 |
| 001100 | vfmv.f.s | - |
| 001101 | vfmv.s.f | - |
| 001110 | - | vfslide1up |
| 001111 | - | vfslide1down |
| 010010 | - | VFUNARY0 |
| 010011 | - | VFUNARY1 |
| 011010 | vmford | - |
| 100010 | VFUNARY0 | - |
| 100011 | VFUNARY1 | - |

1. vadc, vsbc在0.7.1版本中编码为unmasked指令（vm=1）；在0.9版本中编码调整为masked指令（vm=0）。

commit 088d549754cb9ccd7f62428127449f1a00b0b3c4

Merge: f6f3d4b 5a038da

Author: Krste Asanovic <krste@eecs.berkeley.edu>

Date: Mon Nov 11 00:33:05 2019 -0800

Merge pull request #317 from riscv/vmadc-optional-carry-in

Make carry-in optional for vmadc/vmsbc; re-encode vadc/vsbc accordingly

1. vmadc, vmsbc在0.7.1版本中编码为unmasked指令（vm=1）；在0.9版本中编码调整为masked指令（vm=0）和unmasked指令（vm=1）两种都支持。

vmadc and vmsbc add or subtract the source operands, optionally add the carry-in or subtract the borrow-in if masked (vm=0), and write the result back to mask register vd. If unmasked (vm=1), there is no carry-in or borrow-in.