Proposed Instructions for the RISC-V Base P Extension

John Hauser

March 20, 2026

Warning! This document is only a draft proposal and is not an official document of the RISC-V International Association. The Base P extension that is eventually ratified by RISC-V International is liable to differ from this proposal in many details.

This document proposes a set of instructions for the RISC-V Base P extension, both for RV32 and for RV64.

This version (020) differs from the previous one (018) with the addition of:

- SHL instructions (shift left or right, logical) that are unsigned versions of the SHA instructions (shift left or right, arithmetic); and
- for RV64 only, 'P'-suffix versions of PNCLIP instructions that are similar to RV32 PNCLIP instructions without shifting or rounding.

The complete set of instructions added for RV32 are:

PSSHL.HS	SSHL	PSSHL.DHS	PSSHL.DWS
PSSHLR.HS	SSHLR	PSSHLR.DHS	PSSHLR.DWS

And for RV64:

PSSHL.HS	PSSHL.WS	SHL	PNCLIPP.B	PNCLIPP.H	PNCLIPP.W
PSSHLR.HS	PSSHLR.WS	SHLR	PNCLIPUP.B	PNCLIPUP.H	PNCLIPUP.W

The instructions being proposed for the Base P extension are listed here by name, but this document does not, for the most part, describe exactly what each instruction does. Rather:

- Instruction names are constructed based on instruction function, according to a set of rules given by another document, *System for RISC-V P Extension Instruction Names*.
- The correspondences between the instructions here and those of the earlier draft P proposal are detailed in Annex A of this document (a separate PDF).
- For new instructions without a corresponding instruction in the earlier draft P proposal, exact functions are documented in a separate text file.

1 Commonalities

Some general principles apply to the selection of instructions for the Base P extension:

- The default width of operands is XLEN, 32 bits for RV32, or 64 bits for RV64. Where both RV32 and RV64 have the same packed-SIMD instruction, the instruction operates on 32-bit-size operands for RV32, but 64-bit-size operands for RV64, unless indicated specially otherwise.

- An instruction can have no more than three independent register operands: two source operands and one destination. But unlike the base RISC-V ISA, instructions may both read and write their destination operand, which is necessary for accumulate operations, for example.

- Instructions with register-pair operands exist only for RV32, not for RV64. RV32 needs register-pair operands in order to operate conveniently on 64-bit fixed-point values, but RV64 already supports 64-bit values inherently.

  (Although some computations might benefit from specialized RV64 instructions with register-pair operands, the extra costs of implementing register-pair operands in a physical processor are judged not to be justified for only a few, limited uses. For applications that really need 128-bit or wider packed-SIMD operands for performance, product developers are advised to look to the RISC-V V extension.)

- An instruction with register-pair operands (RV32 only) can read from a maximum of one register pair and two single registers, or equivalently, a maximum of two pairs. For example, if an instruction reads from a register-pair destination operand, and both its source operands are also registers, the source operands can be only single registers, not pairs.

  (This rule limits the number of register ports and associated forwarding paths needed in a hardware implementation, when instructions are not broken into multiple micro-ops.)

- There is no concern for overflow of 64-bit values. Hence, there are no instructions that saturate doubleword results, and no instructions that perform averaging addition/subtraction of doublewords.

- For all instructions that accumulate (adding a computed value to the previous value of the destination operand), the accumulate addition does not saturate. If saturation is needed, a separate instruction for saturating addition must be executed.

- No instructions exist that do subtract-accumulate (subtract a computed value from the previous value of the destination operand). If needed, subtraction must be done by a separate instruction.

- For any instruction that includes an accumulate operation, an equivalent instruction without accumulate is usually provided too.

- Among instructions that exist primarily to aid complex arithmetic, only halfword-size and word-size elements are supported, not byte-size, because complex numbers with byte-size components are not considered common enough to justify dedicated instructions. (Unsigned components are also not supported, for obvious reasons.)

Most packed-SIMD instructions that do not multiply have a fairly regular structure, whereas those that do multiply exhibit more variety. The instructions of the Base P extension that do not perform any multiplications are listed in the next section, while instructions that do multiply are grouped together separately in Section 3.

2 Instructions without multiplications

The non-multiplying instructions proposed for the Base P extension are presented in tables in two subsections, the first for instructions that do not have register-pair operands (RV32 and RV64), and then another for instructions with register-pair operands (RV32 only).

The following explains the more unusual or esoteric functions among the proposed instructions:

SH1ADD, SSH1SADD:

Function SSH1SADD (saturating shift left by 1, saturating addition) is a double-saturating version of the Zba extension's SH1ADD instruction. It serves as a building block to better support full-width "Q-format" products (a full-width product shifted left by 1 bit) in combination with saturating addition. While the operation of shifting left by 1 bit and saturating, "SSH1", can be done using SADD, having SSH1SADD available allows two instructions (SADD and SADD) to be reduced to one.

SH1ADD is of course the same computation without any saturation. This can be useful as well for full-width "Q-format" products in circumstances where saturation is not needed or desired.

Instructions for byte-width SH1ADD or SSH1SADD are not proposed because full-width products are always at least 16 bits wide.

PPAIRE, PPAIREO, PPAIROE, PPAIRO:

See: `https://lists.riscv.org/g/tech-p-ext/message/772`
`https://lists.riscv.org/g/tech-p-ext/message/247`

ZIP, UNZIP, WZIP:

See: `https://lists.riscv.org/g/tech-p-ext/message/287`

SLX, SRX:

Shift Left Extended and Shift Right Extended.

See: `https://lists.riscv.org/g/tech-p-ext/message/267`
`https://lists.riscv.org/g/tech-p-ext/message/272`
`https://lists.riscv.org/g/tech-p-ext/message/277`

MVM, MVMN, MERGE:

Move Masked, Move Masked Not, and Merge.

See: `https://lists.riscv.org/g/tech-p-ext/message/267`
`https://lists.riscv.org/g/tech-p-ext/message/276`

WADD, WADDA, WSUB, WSUBA:

For RV32, instructions for widening addition and subtraction, with and without accumulate, are justified as follows: First, addition/subtraction in various forms are among the most common operations of a computer. Second, other widening instructions for RV32, such as WMACC (widening multiply and accumulate), also require the same structure, with single-register source operands, and accumulation into a register-pair destination. And third, much like SH1ADD and SSH1SADD, widening addition/subtraction is useful sometimes as a building block for functions that may exist as single instructions in other architectures but not in the Base P extension for RISC-V.

2.1 No operands are register pairs

B RV32/RV64	H RV32/RV64	W RV32	W RV64	D RV64 only
PLI.B (*1)	PLI.H (*2) PLUI.H	(ADDI) (LUI)	PLI.W (*2) PLUI.W	(ADDI) (*3)
PADD.BS PADD.B PSUB.B PSADD.B PSADDU.B PSSUB.B PSSUBU.B PAADD.B PAADDU.B PASUB.B PASUBU.B	PADD.HS PADD.H PSUB.H PSADD.H PSADDU.H PSSUB.H PSSUBU.H PAADD.H PAADDU.H PASUB.H PASUBU.H	(ADD) (ADD) (SUB) SADD SADDU SSUB SSUBU AADD AADDU ASUB ASUBU	PADD.WS PADD.W PSUB.W PSADD.W PSADDU.W PSSUB.W PSSUBU.W PAADD.W PAADDU.W PASUB.W PASUBU.W	(ADD) (ADD) (SUB) (*4)
(*5)	PSH1ADD.H PSSH1SADD.H	(SH1ADD) SSH1SADD	PSH1ADD.W PSSH1SADD.W	(SH1ADD) (*4)
(*6)	PAS.HX PSA.HX PSAS.HX PSSA.HX PAAS.HX PASA.HX	*not applicable* (*7)	PAS.WX PSA.WX PSAS.WX PSSA.WX PAAS.WX PASA.WX	*not applicable* (*8)
PABD.B PABDU.B PSABS.B	PABD.H PABDU.H PSABS.H	(*9) (*10)		(*9) (*4)
PREDSUM.BS PREDSUMU.BS	PREDSUM.HS PREDSUMU.HS	(ADD) (ADD)	PREDSUM.WS PREDSUMU.WS	(ADD) (ADD)
PABDSUMU.B PABDSUMAU.B	(*11)			

(*1) PLI.B (Packed-SIMD Load Immediate, Byte elements) can load any 8-bit immediate into all destination bytes.

(*2) The immediates that can be loaded into halfword and word elements are 10 bits wide, signed. PLI (Packed-SIMD Load Immediate) and PLUI (Packed-SIMD Load Upper Immediate) differ in whether the immediate is loaded into the least-significant or most-significant 10 bits of each destination element.

(*3) Use existing instructions for loading doubleword constants.

(*4) Saturation is not supported at doubleword size, nor averaging add/subtract.

(*5) Operations SH1ADD and SSH1SADD are not supported for bytes.

(*6) The instructions that add/subtract even/odd elements (PAS.HX, PSA.HX, etc.) exist for complex arithmetic, which is not equally supported for byte-size components.

(*7) A single RV32 register can hold only one 32-bit word, so cannot contain two elements to swap. The equivalent function is performed by two scalar instructions, such as ADD and SUB.

(*8) A single RV64 register can hold only one 64-bit doubleword, so cannot contain two elements to swap. The equivalent function is performed by two scalar instructions, such as ADD and SUB.

(*9) Absolute difference can be computed in three operations: MAX, MIN, and SUB for ABD; or MAXU, MINU, and SUB for ABDU. Absolute value can be computed either with the scalar ABS instruction or in two operations: SUB to negate, then MINU of the original and negated values.

(*10) Saturating absolute value SABS can be computed in two operations: SSUB to negate with saturation, then MINU of the original and negated values.

(*11) Sum of absolute difference is not supported for sizes larger than bytes.

B	H	W		D
RV32/RV64	RV32/RV64	RV32	RV64	RV64 only
	PSEXT.H.B	(SEXT.B)	PSEXT.W.B	(SEXT.B)
		(SEXT.H)	PSEXT.W.H	(SEXT.H)
(*1)	PSATI.H	SATI	PSATI.W	SATI
	PUSATI.H	USATI	PUSATI.W	USATI
PSLLI.B	PSLLI.H	(SLLI)	PSLLI.W	(SLLI)
PSLL.BS	PSLL.HS	(SLL)	PSLL.WS	(SLL)
PSRLI.B	PSRLI.H	(SRLI)	PSRLI.W	(SRLI)
PSRL.BS	PSRL.HS	(SRL)	PSRL.WS	(SRL)
PSRAI.B	PSRAI.H	(SRAI)	PSRAI.W	(SRAI)
PSRA.BS	PSRA.HS	(SRA)	PSRA.WS	(SRA)
(*1)	PSSHL.HS	SSHL	PSSHL.WS	SHL (*2)
	PSSHLR.HS	SSHLR	PSSHLR.WS	SHLR
	PSSLAI.H	SSLAI	PSSLAI.W	(*2)
	PSRARI.H	SRARI	PSRARI.W	SRARI
	PSSHA.HS	SSHA	PSSHA.WS	SHA (*2)
	PSSHAR.HS	SSHAR	PSSHAR.WS	SHAR
PMIN.B	PMIN.H	(MIN)	PMIN.W	(MIN)
PMINU.B	PMINU.H	(MINU)	PMINU.W	(MINU)
PMAX.B	PMAX.H	(MAX)	PMAX.W	(MAX)
PMAXU.B	PMAXU.H	(MAXU)	PMAXU.W	(MAXU)
PMSEQ.B	PMSEQ.H	MSEQ	PMSEQ.W	(*3)
PMSLT.B	PMSLT.H	MSLT	PMSLT.W	
PMSLTU.B	PMSLTU.H	MSLTU	PMSLTU.W	

(*1) These operations are not supported for bytes: arbitrary-width saturation (SAT, USAT) and advanced shifts (SSHL, SSHLR, SSLA, SRAR, SSHA, and SSHAR).

(*2) Saturation is not supported at doubleword size.

(*3) Condition masks are not supported at doubleword size. Use existing instructions for conditional execution, or negate the result of a standard SLT or SLTU to form a 64-bit mask.

B		H		W
RV32	RV64	RV32	RV64	RV64 only
PPAIRE.B		(PACK) (*1)	PPAIRE.H	(PACK) (*2)
PPAIREO.B		PPAIREO.H		PPAIREO.W
PPAIROE.B		PPAIROE.H		PPAIROE.W
PPAIRO.B		PPAIRO.H		PPAIRO.W
(REV8)		(*3)	REV16	(*4)
(*5)	ZIP8P	(PACK) (*5)	ZIP16P	(PACK)
	ZIP8HP	PPAIRO.H	ZIP16HP	PPAIRO.W
(*6)	UNZIP8P	(PACK)	UNZIP16P	(PACK)
	UNZIP8HP	PPAIRO.H	UNZIP16HP	PPAIRO.W
(*7)	PNCLIPP.B	(*7)	PNCLIPP.H	PNCLIPP.W
	PNCLIPUP.B		PNCLIPUP.H	PNCLIPUP.W

(*1) For RV32, PPAIRE.H is a pseudoinstruction for PACK.

(*2) For RV64, PPAIRE.W is a pseudoinstruction for PACK.

(*3) The two halfwords of a word can be swapped using PPAIROE.H.

(*4) The two words of a doubleword can be swapped using PPAIROE.W.

(*5) For RV32, the equivalent of both ZIP8P and ZIP8HP can be done together by a single WZIP8P instruction, and the equivalent of both ZIP16P and ZIP16HP can be done together by a single WZIP16P instruction.

(*6) For RV32, the functions of UNZIP8P and UNZIP8HP can be achieved by instruction PNSRLI.B (narrowing shift-right), with a shift distance of 0 for UNZIP8P or 8 for UNZIP8HP.

(*7) RV32 has more capable PNCLIP instructions with an even-odd register pair as the first source operand. The 'P'-suffix versions of PNCLIP exist for RV64 because the P extension for RV64 has no instructions with register-pair operands.

scalar only			
RV32/RV64			RV64
ABS	SLX	MVM	ABSW
(CLZ)	SRX	MVMN	(CLZW)
CLS		MERGE	CLSW
REV			

2.2 One or more operands are even-odd register pairs (RV32 only)

RV32 only, register-pair destination		
B→H	H→W	W→D
PWADD.B	PWADD.H	WADD
PWADDA.B	PWADDA.H	WADDA
PWADDU.B	PWADDU.H	WADDU
PWADDAU.B	PWADDAU.H	WADDAU
PWSUB.B	PWSUB.H	WSUB
PWSUBA.B	PWSUBA.H	WSUBA
PWSUBU.B	PWSUBU.H	WSUBU
PWSUBAU.B	PWSUBAU.H	WSUBAU
PWSLLI.B	PWSLLI.H	WSLLI
PWSLL.BS	PWSLL.HS	WSLL
PWSLAI.B	PWSLAI.H	WSLAI
PWSLA.BS	PWSLA.HS	WSLA

RV32 only, register-pair destination		
B	H	W
WZIP8P	WZIP16P	(*1)

(*1) For RV32, the equivalent function for words is accomplished by two MV instructions.

RV32 only, register-pair operands			
B	H	W	D
PLI.DB	PLI.DH PLUI.DH	(*1)	(*2)
PADD.DB PSUB.DB PSADD.DB PSADDU.DB PSSUB.DB PSSUBU.DB PAADD.DB PAADDU.DB PASUB.DB PASUBU.DB	PADD.DH PSUB.DH PSADD.DH PSADDU.DH PSSUB.DH PSSUBU.DH PAADD.DH PAADDU.DH PASUB.DH PASUBU.DH	PADD.DW PSUB.DW PSADD.DW PSADDU.DW PSSUB.DW PSSUBU.DW PAADD.DW PAADDU.DW PASUB.DW PASUBU.DW	ADDD SUBD
(*3)	PSH1ADD.DH PSSH1SADD.DH	PSH1ADD.DW PSSH1SADD.DW	
(*4)	PAS.DHX PSA.DHX PSAS.DHX PSSA.DHX PAAS.DHX PASA.DHX	(*5)	*not applicable* (*6)
PABD.DB PABDU.DB PSABS.DB	PABD.DH PABDU.DH PSABS.DH	(*7) (*8)	(*2)

(*1) PLI.DW and PLUI.DW would be inconvenient to support because RV32 has no exactly corresponding scalar instruction for words.

(*2) Few doubleword scalar instructions are provided for RV32. In particular, saturation is not supported at doubleword size, nor averaging add/subtract.

(*3) Operations SH1ADD and SSH1SADD are not supported for bytes.

(*4) The instructions that add/subtract even/odd elements exist for complex arithmetic, which is not equally supported for byte-size components.

(*5) For RV32, swapping words (.DWX) is not supported. The equivalent function is performed by two scalar instructions, such as ADD and SUB.

(*6) An RV32 register pair can hold only one 64-bit doubleword, so cannot contain two elements to swap.

(*7) Absolute difference can be computed in three operations: MAX, MIN, and SUB for ABD; or MAXU, MINU, and SUB for ABDU. Absolute values of two words can be done with two ABS instructions.

(*8) Saturating absolute value SABS can be computed in two operations: SSUB to negate with saturation, then MINU of the original and negated values.

RV32 only, register-pair first source (only)		
B	H	W
PREDSUM.DBS PREDSUMU.DBS	PREDSUM.DHS PREDSUMU.DHS	(*1)

(*1) The equivalent function is accomplished by two ADD instructions, or a single WADDA or WADDAU.

RV32 only, register-pair operands			
B	H	W	D
	PSEXT.DH.B	PSEXT.DW.B	(*1)
		PSEXT.DW.H	
(*2)	PSATI.DH	PSATI.DW	
	PUSATI.DH	PUSATI.DW	
PSLLI.DB	PSLLI.DH	PSLLI.DW	
PSRLI.DB	PSRLI.DH	PSRLI.DW	
PSRAI.DB	PSRAI.DH	PSRAI.DW	
(*2)	PSSLAI.DH	PSSLAI.DW	
	PSRARI.DH	PSRARI.DW	
PMIN.DB	PMIN.DH	PMIN.DW	
PMINU.DB	PMINU.DH	PMINU.DW	
PMAX.DB	PMAX.DH	PMAX.DW	
PMAXU.DB	PMAXU.DH	PMAXU.DW	
PMSEQ.DB	PMSEQ.DH	PMSEQ.DW	
PMSLT.DB	PMSLT.DH	PMSLT.DW	
PMSLTU.DB	PMSLTU.DH	PMSLTU.DW	

(*1) Few doubleword scalar instructions are provided for RV32. Doubleword shifts by a constant distance can be done in two instructions, making use of narrowing shift instructions NSRLI and NSRAI.

(*2) These operations are not supported for bytes: arbitrary-width saturation (SAT, USAT) and advanced shifts (SSLA, SRAR).

RV32 only, register-pair first source and destination			
B	H	W	D
PADD.DBS	PADD.DHS	PADD.DWS	(*1)
PSLL.DBS	PSLL.DHS	PSLL.DWS	
PSRL.DBS	PSRL.DHS	PSRL.DWS	
PSRA.DBS	PSRA.DHS	PSRA.DWS	
(*2)	PSSHL.DHS	PSSHL.DWS	
	PSSHLR.DHS	PSSHLR.DWS	
	PSSHA.DHS	PSSHA.DWS	
	PSSHAR.DHS	PSSHAR.DWS	

(*1) Few doubleword scalar instructions are provided for RV32.

(*2) Advanced shifts are not supported for bytes.

RV32 only, register-pair operands		
B	H	W
PPAIRE.DB	PPAIRE.DH	(*1)
PPAIREO.DB	PPAIREO.DH	
PPAIROE.DB	PPAIROE.DH	
PPAIRO.DB	PPAIRO.DH	

(*1) Packing two words into a doubleword is accomplished by two MV instructions.

RV32 only, register-pair first source (only)		
H→B	W→H	D→W
PNSRLI.B	PNSRLI.H	NSRLI
PNSRL.BS	PNSRL.HS	NSRL
PNSRAI.B	PNSRAI.H	NSRAI
PNSRA.BS	PNSRA.HS	NSRA
PNSRARI.B	PNSRARI.H	NSRARI
PNSRAR.BS	PNSRAR.HS	NSRAR
PNCLIPI.B	PNCLIPI.H	NCLIPI
PNCLIP.BS	PNCLIP.HS	NCLIP
PNCLIPRI.B	PNCLIPRI.H	NCLIPRI
PNCLIPR.BS	PNCLIPR.HS	NCLIPR
PNCLIPIU.B	PNCLIPIU.H	NCLIPIU
PNCLIPU.BS	PNCLIPU.HS	NCLIPU
PNCLIPRIU.B	PNCLIPRIU.H	NCLIPRIU
PNCLIPRU.BS	PNCLIPRU.HS	NCLIPRU

3 Instructions that perform multiplications

Besides the more general principles listed earlier, some additional ones apply specifically to the selection of multiplication instructions for the Base P extension:

- For byte × byte multiplication, only full-width products (B × B → H) are supported directly by instructions, not byte-width products (B × B → B), because the latter do not maintain enough precision for most computations. Algorithms that actually need byte-size products must convert back from full-width, halfword-size products.
- For byte × byte multiplication, multiply-add instructions are provided only in forms that do four-element-wide horizontal addition (4ADD), as these are believed to be sufficient for most needs. For any other cases, byte multiplication and subsequent addition can always be done using separate instructions.
- Instructions that multiply doublewords are not provided by the P extension.
- RV32 instructions with register-pair operands may not take more than 32 bits of factors to multiply from each of two source operands—i.e., 32 bits from the first source operand, and 32 bits from the second source operand. This limits the number of bits that may be multiplier inputs. (When more multiplier capability is needed for performance, developers are expected to step up to RV64P systems, or switch to the V extension.)

As before, the first subsection below presents the instructions that do not have register-pair operands, followed by a separate subsection for RV32-only instructions with register-pair destinations.

For instructions listed with suffix '.Bn' or '.Bnn', each n is the number of the byte sub-element to take from each halfword of the respective source operand, either '0' for bits 7:0 or '1' for bits 15:8 of each halfword. Likewise, for instructions listed with suffix '.Hn' or '.Hnn', each n is the number of the halfword sub-element to take from each word of the respective source operand, either '0' for bits 15:0 or '1' for bits 31:16 of each word. And for instructions listed with suffix '.Wnn', each n is the number of the word sub-element to take from the respective doubleword source operand, either '0' for bits 31:0 or '1' for bits 63:32. If there is only one digit, it applies to the second source operand. Where there are two digits, the only combinations allowed are '00', '01', and '11'. For signed-unsigned (SU) multiplications, '01' is not supported, so the only combinations allowed are '00' and '11'.

3.1 No operands are register pairs

B×B→B	H×H→H	W×W→W	
RV32/RV64	RV32/RV64	RV32	RV64
(*1)	PMULH.H	(MULH)	PMULH.W
	PMULHR.H	MULHR	PMULHR.W
	PMULHSU.H	(MULHSU)	PMULHSU.W
	PMULHRSU.H	MULHRSU	PMULHRSU.W
	PMULHU.H	(MULHU)	PMULHU.W
	PMULHRU.H	MULHRU	PMULHRU.W
	PMULQ.H	MULQ	PMULQ.W
	PMULQR.H	MULQR	PMULQR.W
	PMHACC.H	MHACC	PMHACC.W
	PMHRACC.H	MHRACC	PMHRACC.W
	PMHACCSU.H	MHACCSU	PMHACCSU.W
	PMHRACCSU.H	MHRACCSU	PMHRACCSU.W
	PMHACCU.H	MHACCU	PMHACCU.W
	PMHRACCU.H	MHRACCU	PMHRACCU.W

(*1) For byte × byte multiplication, instructions are provided only for full-width products (B×B→H), not byte-width products (B×B→B).

B×B→B→H	H×H→H→W		W×W→W→D
RV32/RV64	RV32	RV64	RV64 only
(*1)	MQACC.H*nn*	PMQACC.W.H*nn*	MQACC.W*nn*
	MQRACC.H*nn*	PMQRACC.W.H*nn*	MQRACC.W*nn*
	PMQ2ADD.H		PMQ2ADD.W
	PMQ2ADDA.H		PMQ2ADDA.W
	PMQR2ADD.H		PMQR2ADD.W
	PMQR2ADDA.H		PMQR2ADDA.W

(*1) For byte × byte multiplication, instructions are provided only for full-width products (B×B→H), not byte-width products (B×B→B).

B×B→H RV32/RV64	H×H→W RV32	 RV64	W×W→D RV64 only
PMUL.H.B*nn* PMULSU.H.B*nn* PMULU.H.B*nn*	MUL.H*nn* MULSU.H*nn* MULU.H*nn*	PMUL.W.H*nn* PMULSU.W.H*nn* PMULU.W.H*nn*	MUL.W*nn* MULSU.W*nn* MULU.W*nn*
(*1)	MACC.H*nn* MACCSU.H*nn* MACCU.H*nn*	PMACC.W.H*nn* PMACCSU.W.H*nn* PMACCU.W.H*nn*	MACC.W*nn* MACCSU.W*nn* MACCU.W*nn*
	PM2ADD.H PM2ADDA.H PM2ADDSU.H PM2ADDASU.H PM2ADDU.H PM2ADDAU.H PM2ADD.HX PM2ADDA.HX PM2SADD.H (*2) PM2SADD.HX PM2SUB.H PM2SUBA.H PM2SUB.HX PM2SUBA.HX		PM2ADD.W PM2ADDA.W PM2ADDSU.W PM2ADDASU.W PM2ADDU.W PM2ADDAU.W PM2ADD.WX PM2ADDA.WX (*3) PM2SUB.W PM2SUBA.W PM2SUB.WX PM2SUBA.WX

(*1) For byte × byte multiplication, the only multiply-add instructions provided are those with four-element-wide horizontal addition (4ADD).

(*2) Saturation of the horizontal addition (2SADD) will occur only when all four source halfwords equal −`0x8000`. Versions of these instructions that accumulate are not provided because saturation is not supported for an accumulate operation. To get the effect of multiply-accumulate, a PM2SADD.H or PM2SADD.HX will most likely be combined with SADD or PSADD.DW (RV32) or PSADD.W (RV64).

(*3) Saturation is not supported at doubleword size.

B×B→H→W RV32/RV64	H×H→W→D RV64 only
PM4ADD.B	PM4ADD.H
PM4ADDA.B	PM4ADDA.H
PM4ADDSU.B	PM4ADDSU.H
PM4ADDASU.B	PM4ADDASU.H
PM4ADDU.B	PM4ADDU.H
PM4ADDAU.B	PM4ADDAU.H

H×B→H RV32/RV64	W×H→W RV32	 RV64
PMULH.H.B*n*	MULH.H*n*	PMULH.W.H*n*
PMULHSU.H.B*n*	MULHSU.H*n*	PMULHSU.W.H*n*
PMHACC.H.B*n*	MHACC.H*n*	PMHACC.W.H*n*
PMHACCSU.H.B*n*	MHACCSU.H*n*	PMHACCSU.W.H*n*

3.2 Destination is an even-odd register pair (RV32 only)

RV32 only, register-pair destination		
B×B→B→H	H×H→H→W	W×W→W→D
(*1)	PMQWACC.H	MQWACC
	PMQRWACC.H	MQRWACC

(*1) For byte × byte multiplication, instructions are provided only for full-width products (B×B→H), not byte-width products (B×B→B).

RV32 only, register-pair destination		
B×B→H	H×H→W	W×W→D
PWMUL.B	PWMUL.H	WMUL
PWMULSU.B	PWMULSU.H	WMULSU
PWMULU.B	PWMULU.H	WMULU
(*1)	PWMACC.H	WMACC
	PWMACCSU.H	WMACCSU
	PWMACCU.H	WMACCU

(*1) For byte × byte multiplication, the only multiply-add instructions provided are those with four-element-wide horizontal addition (4ADD).

RV32 only, register-pair destination	
B×B→H→W	H×H→W→D
(*1)	PM2WADD.H
	PM2WADDA.H
	PM2WADDSU.H
	PM2WADDASU.H
	PM2WADDU.H
	PM2WADDAU.H
	PM2WADD.HX
	PM2WADDA.HX
	PM2WSUB.H
	PM2WSUBA.H
	PM2WSUB.HX
	PM2WSUBA.HX

(*1) For byte × byte multiplication, the only multiply-add instructions provided are those with four-element-wide horizontal addition (4ADD).