

Proposed Instruction Encodings for the RISC-V Base P Extension

John Hauser

February 18, 2025

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 encodings for the instructions of the RISC-V Base P extension, both for RV32 and for RV64. For more about this proposal, see my other documents:

- *System for RISC-V P Extension Instruction Names*
- *Proposed Instructions for the RISC-V Base P Extension*

This version (014) differs from the previous one (012) by adopting a modified system for naming instructions, causing these instruction names to change:

all *.B.B0	→	*.BS	all *.DB.B0	→	*.DBS
all *.H.H0	→	*.HS	all *.DH.H0	→	*.DHS
all *.W.W0	→	*.WS	all *.DW.W	→	*.DWS
PREDSUM.*	→	PREDSUM.*S			
PREDSUMU.*	→	PREDSUMU.*S			
PSEXTB.H	→	PSEXT.H.B	PSEXTB.DH	→	PSEXT.DH.B
PSEXTB.W	→	PSEXT.W.B	PSEXTB.DW	→	PSEXT.DW.B
PSEXT.H.W	→	PSEXT.W.H	PSEXT.H.DW	→	PSEXT.DW.H
all *.H.BEE	→	*.H.B00	all *.H.BE	→	*.H.B0
all *.H.BEO	→	*.H.B01	all *.H.BO	→	*.H.B1
all *.H.BOO	→	*.H.B11			
all *.W.HEE	→	*.W.H00	all *.W.HE	→	*.W.H0
all *.W.HEO	→	*.W.H01	all *.W.HO	→	*.W.H1
all *.W.HOO	→	*.W.H11			

1 Instruction formats

A few non-SIMD instructions of the P extension (including SH1ADD, ABS, SEXT.B, SEXT.H, CLZ, CLS, MIN, MINU, MAX, MAXU, PACK, REV8, and REV16) are encoded alongside other ordinary RISC-V “scalar” instructions in the major opcodes known as OP (binary code 0110011), OP-IMM (0010011), and OP-IMM-32 (0011011).

All other Base P instructions are encoded either: (a) within major opcode OP-32 (binary code 0111011) with instruction bit 31 = 1; or (b) within OP-IMM-32 with bits 14:12 (func3) = binary 010, 100, or 110.

Apart from the subset of non-SIMD instructions already mentioned plus a few other special cases, the following instruction formats are used for all Base P instructions that have no register-pair operands, only single registers:

32	27	20	15	12	7	0		
1	x x x 0	w-uimm	rs1	0 1 0	rd	0 0 1 1 0 1 1		
32	27	25	20	15	12	7	0	
1	x x x 1	w	rs2	rs1	0 1 0	rd	0 0 1 1 0 1 1	
32	27	20	15	12	7	0		
1	x x x 0	w-uimm	rs1	1 0 0	rd	0 0 1 1 0 1 1		
32	27	25	20	15	12	7	0	
1	x x x 1	w	rs2	rs1	1 0 0	rd	0 0 1 1 0 1 1	
32	27	25	20	15	12	7	0	
1	x x x x	w	rs2	rs1	x x 0	rd	0 1 1 1 0 1 1	
32	28	27	25	20	15	12	7	0
1	x x x	R	w	rs2	rs1	x x 1	rd	0 1 1 1 0 1 1

Note that the first four formats are in major opcode OP-IMM-32, and the last two are in OP-32. Of the OP-IMM-32 formats, those with func3 = binary 010 are mostly shifts left, while those with func3 = binary 100 are mostly shifts right.

The two-bit field w often selects the SIMD element width, whether bytes, halfwords, or words, although that is not its only function. The most common, though not universal, encoding for w has binary 00 = halfwords, 01 = words, 10 = bytes, and 11 = a doubleword. Field w-uimm encodes both an element width and an unsigned immediate operand whose size varies with the element width; for example, 3 bits for bytes, 4 bits for halfwords, 5 bits for words, and 6 bits for a doubleword.

For the last format, when bit R = 1, an instruction both reads and writes the destination operand. Usually this is done for an accumulate operation, although there are some other instances, such as SRX (Shift Right Extended) and MVM (Move Masked). Instructions encoded in the other formats listed above never read the destination.

With RV32, the P extension has additional instruction formats for the cases when operands are even-odd register pairs:

32	27	20	15	12	8	0		
0	x x x 0	w-uimm	rs1	0 1 0	rd _p	0 0 0 1 1 0 1 1		
32	27	25	20	15	12	8	0	
0	x x x 1	w	rs2	rs1	0 1 0	rd _p	0 0 0 1 1 0 1 1	
32	28	27	25	20	15	12	8	0
0	x x x	R	w	rs2	rs1	0 1 0	rd _p	1 0 0 1 1 0 1 1
32	27	20	16	12	7	0		
0	x x x 0	w-uimm	rs1 _p	1 1 0 0	rd	0 0 1 1 0 1 1		
32	27	25	20	16	12	7	0	
0	x x x 1	w	rs2	rs1 _p	x 1 0 0	rd	0 0 1 1 0 1 1	
32	27	20	16	12	8	0		
0	x x x 0	w-uimm	rs1 _p	x 1 1 0	rd _p	0 0 0 1 1 0 1 1		
32	27	25	20	16	12	8	0	
0	x x x 1	w	rs2	rs1 _p	x 1 1 0	rd _p	0 0 0 1 1 0 1 1	
32	27	25	21	20	16	12	8	0
1	x x x x	w	rs2 _p	x	rs1 _p	x 1 1 0	rd _p	0 0 0 1 1 0 1 1

All of these are in major opcode OP-IMM-32. The formats here with func3 = binary 010 primarily encode widening shifts left and other widening instructions, and those with func3 = binary 100 encode narrowing shifts right and reductions. The last three formats, with func3 = binary 110, encode all the double-wide instructions, which have at least the first source and destination as register pairs, and often the second source operand as well.

The third format listed above, with an R bit, encodes widening adds, subtracts, and multiplies that may optionally read the destination for an accumulate operation (WADDA, WSUBA, WMACC, etc.).

The following format is not used by Base P but is recommended for a Zp extension that adds double-wide multiply instructions:*

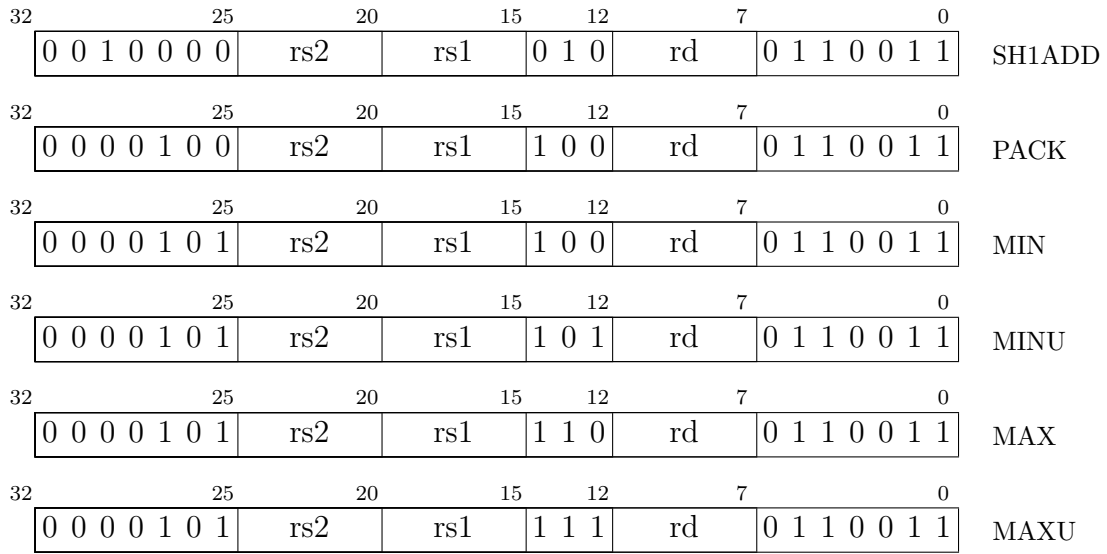
32	28	27	25	21	20	16	12	8	0
1	x x x	R	w	rs2 _p	x	rs1 _p	x 1 1 0	rd _p	1 0 0 1 1 0 1 1

2 Encodings of instructions with no register-pair operands

In this section, all instruction tables show separately the allocations of instructions for RV32 and for RV64. Asterisks (*) mark where there are differences between the RV32 and RV64 halves of a table.

Where an instruction name is within braces { }, there is no such instruction proposed for the Base P extension, but the indicated encoding is recommended should the instruction be provided by another extension.

32	27	20	15	12	7	0	
0	1	1	0	0	0	0	0 0 1 0 0 1 1
			rs1		rd		CLZ
32	27	20	15	12	7	0	
0	1	1	0	0	0	0	0 0 1 0 0 1 1
			rs1		rd		CLS
32	27	20	15	12	7	0	
0	1	1	0	0	0	1	0 0 1 0 0 1 1
			rs1		rd		SEXT.B
32	27	20	15	12	7	0	
0	1	1	0	0	0	1	0 0 1 0 0 1 1
			rs1		rd		SEXT.H
32	27	20	15	12	7	0	
0	1	1	0	0	0	1	1 1 0 0 1 1 1
			rs1		rd		ABS
32	27	20	15	12	7	0	
0	1	1	0	1	0	0	1 0 1 0 0 1 1
			rs1		rd		REV8 (RV32)
32	27	20	15	12	7	0	
0	1	1	0	1	0	0	1 1 1 1 1 1 1
			rs1		rd		REV (RV32)
32	27	20	15	12	7	0	
0	1	1	0	1	0	1	0 0 0 0 0 1 1
			rs1		rd		REV16 (RV64)
32	27	20	15	12	7	0	
0	1	1	0	1	0	1	0 0 0 0 0 1 1
			rs1		rd		REV8 (RV64)
32	27	20	15	12	7	0	
0	1	1	0	1	0	1	1 1 1 1 1 1 1
			rs1		rd		REV (RV64)
32	27	20	15	12	7	0	
0	1	1	0	0	0	0	0 0 1 1 0 1 1
			rs1		rd		CLZW (RV64)
32	27	20	15	12	7	0	
0	1	1	0	0	0	0	0 0 1 1 0 1 1
			rs1		rd		CLSW (RV64)
32	27	20	15	12	7	0	
0	1	1	0	0	0	0	0 0 1 1 0 1 1
			rs1		rd		ABSW (RV64)



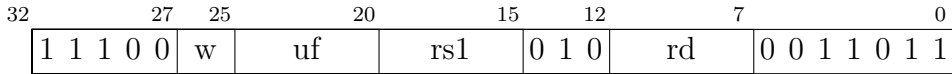
32	31	28	27	20	15	12	7	0
1	f	0	w-uimm	rs1	0 1 0	rd	0 0 1 1 0 1 1	

RV32	w-uimm							
f	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx			
000		PSLLI.B	PSLLI.H	*				
001		{ PSSLLI.B }	{ PSSLLI.H }	* { SLLI }				
010								
011	<i>PLI instructions</i>							
100								
101		{ PSSLAI.B }	PSSLAI.H	* SSLAI				
110	<i>unary instructions</i>							
111	<i>PLUI instructions</i>							
RV64	w-uimm							
f	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx			
000		PSLLI.B	PSLLI.H	* PSLLI.W				
001		{ PSSLLI.B }	{ PSSLLI.H }	* { PSSLLI.W }				
010								
011	<i>PLI instructions</i>							
100								
101		{ PSSLAI.B }	PSSLAI.H	* PSSLAI.W				
110	<i>unary instructions</i>							
111	<i>PLUI instructions</i>							

32	27	25	15	12	7	0
1 0 1 1 0	0 0	imm[8:0 9]	0 1 0	rd	0 0 1 1 0 1 1	PLI.H

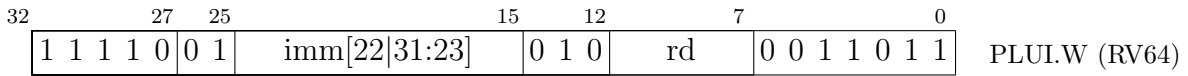
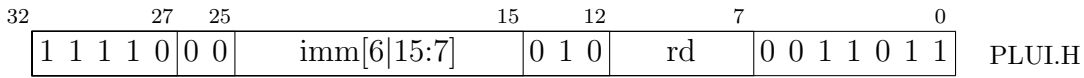
32	27	25	15	12	7	0
1 0 1 1 0	0 1	imm[8:0 9]	0 1 0	rd	0 0 1 1 0 1 1	PLI.W (RV64)

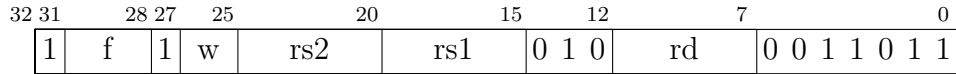
32	27	24	16	12	7	0
1 0 1 1 0	1 0 0	imm[7:0]	0 0 1 0	rd	0 0 1 1 0 1 1	PLI.B



RV32	w			
uf	00	01	10	11
00000	{ PCLZ.H }	*	{ PCLZ.B }	
00001				
00010				
00011	{ PCLS.H }	*	{ PCLS.B }	
00100	PSEXT.H.B	*		
00101		*		
00110				
00111	PSABS.H	* { SABS }	PSABS.B	
01000				
01001				
01010				
01011				
...				
11111				

RV64	w			
uf	00	01	10	11
00000	{ PCLZ.H }	* { PCLZ.W }	{ PCLZ.B }	
00001				
00010				
00011	{ PCLS.H }	* { PCLS.W }	{ PCLS.B }	
00100	PSEXT.H.B	* PSEXT.W.B		
00101		* PSEXT.W.H		
00110				
00111	PSABS.H	* { PSABS.W }	PSABS.B	
01000				
01001				
01010				
01011				
...				
11111				





RV32		w			
f		00	01	10	11
000	PSLL.HS	*	PSLL.BS		
001	PADD.HS	*	PADD.BS		
010	{ PSSHL.HS }	* { SSSL }	{ PSSHL.BS }	*	
011	{ PSSHLR.HS }	* { SSSLR }	{ PSSHLR.BS }	*	
100					
101					
110	PSSHA.HS	* SSHA	{ PSSHA.BS }	*	
111	PSSHAR.HS	* SSHAR	{ PSSHAR.BS }	*	
RV64		w			
f		00	01	10	11
000	PSLL.HS	* PSLL.WS	PSLL.BS		
001	PADD.HS	* PADD.WS	PADD.BS		
010	{ PSSHL.HS }	* { PSSHL.WS }	{ PSSHL.BS }	* { SHL }	
011	{ PSSHLR.HS }	* { PSSHLR.WS }	{ PSSHLR.BS }	* { SHLR }	
100					
101					
110	PSSHA.HS	* PSSHA.WS	{ PSSHA.BS }	* SHA	
111	PSSHAR.HS	* PSSHAR.WS	{ PSSHAR.BS }	* SHAR	

32	31	28	27	20	15	12	7	0
1	f	0	w-uimm	rs1	1 0 0	rd	0 0 1 1 0 1 1	

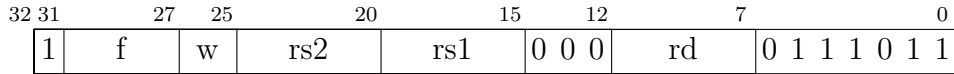
RV32	w-uimm				
f	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSRLI.B	PSRLI.H	*	
001		{ PSRLRI.B }	{ PSRLRI.H }	* { SRLRI }	*
010		{ PUSATI.B }	PUSATI.H	* USATI	*
011					
100		PSRAI.B	PSRAI.H	*	
101		{ PSRARI.B }	PSRARI.H	* SRARI	*
110		{ PSATI.B }	PSATI.H	* SATI	*
111					

RV64	w-uimm				
f	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSRLI.B	PSRLI.H	* PSRLI.W	
001		{ PSRLRI.B }	{ PSRLRI.H }	* { PSRLRI.W }	* { SRLRI }
010		{ PUSATI.B }	PUSATI.H	* PUSATI.W	* USATI
011					
100		PSRAI.B	PSRAI.H	* PSRAI.W	
101		{ PSRARI.B }	PSRARI.H	* PSRARI.W	* SRARI
110		{ PSATI.B }	PSATI.H	* PSATI.W	* SATI
111					

32	31	28	27	25	20	15	12	7	0
1	f	1	w	rs2	rs1	1 0 0	rd	0 0 1 1 0 1 1	

RV32	w			
f	00	01	10	11
000	PSRL.HS	*	PSRL.BS	
001	PREDSUM.HS	*	PREDSUM.BS	
010				
011	PREDSUMU.HS	*	PREDSUMU.BS	
100	PSRA.HS	*	PSRA.BS	
101				
110				
111				

RV64	w			
f	00	01	10	11
000	PSRL.HS	* PSRL.WS	PSRL.BS	
001	PREDSUM.HS	* PREDSUM.WS	PREDSUM.BS	
010				
011	PREDSUMU.HS	* PREDSUMU.WS	PREDSUMU.BS	
100	PSRA.HS	* PSRA.WS	PSRA.BS	
101				
110				
111				



RV32 f	w			
	00	01	10	11
0000	PADD.H	*	PADD.B	
0001				
0010	PSADD.H	* SADD	PSADD.B	
0011	PAADD.H	* AADD	PAADD.B	
0100				
0101				
0110	PSADDU.H	* SADDU	PSADDU.B	
0111	PAADDU.H	* AADDU	PAADDU.B	
1000	PSUB.H	*	PSUB.B	
1001	PDIF.H	* {DIF}	PDIF.B	
1010	PSSUB.H	* SSUB	PSSUB.B	
1011	PASUB.H	* ASUB	PASUB.B	
1100				
1101	PDIFU.H	* {DIFU}	PDIFU.B	
1110	PSSUBU.H	* SSUBU	PSSUBU.B	
1111	PASUBU.H	* ASUBU	PASUBU.B	
RV64 f	w			
	00	01	10	11
0000	PADD.H	* PADD.W	PADD.B	
0001				
0010	PSADD.H	* PSADD.W	PSADD.B	
0011	PAADD.H	* PAADD.W	PAADD.B	
0100				
0101				
0110	PSADDU.H	* PSADDU.W	PSADDU.B	
0111	PAADDU.H	* PAADDU.W	PAADDU.B	
1000	PSUB.H	* PSUB.W	PSUB.B	
1001	PDIF.H	* {PDIF.W}	PDIF.B	
1010	PSSUB.H	* PSSUB.W	PSSUB.B	
1011	PASUB.H	* PASUB.W	PASUB.B	
1100				
1101	PDIFU.H	* {PDIFU.W}	PDIFU.B	
1110	PSSUBU.H	* PSSUBU.W	PSSUBU.B	
1111	PASUBU.H	* PASUBU.W	PASUBU.B	

32 31	27	25	20	15	12	7	0
1	f	w	rs2	rs1	0 0 1	rd	0 1 1 1 0 1 1

RV32 f	w			
	00	01	10	11
0000 0001 0010 0011	PMUL.H.B01 { PMACC.H.B01 }	* MUL.H01 * MACC.H01		SLX * *
0100 0101 0110 0111	MVM PMULU.H.B01 { PMACCU.H.B01 }	MVMN * MULU.H01 * MACCU.H01	MERGE PDIFSUMU.B PDIFSUMAU.B	SRX * *
1000 1001 1010 1011				
1100 1101 1110 1111				
RV64 f	w			
	00	01	10	11
0000 0001 0010 0011	PMUL.H.B01 { PMACC.H.B01 }	* PMUL.W.H01 * PMACC.W.H01		SLX * MUL.W01 * MACC.W01
0100 0101 0110 0111	MVM PMULU.H.B01 { PMACCU.H.B01 }	MVMN * PMULU.W.H01 * PMACCU.W.H01	MERGE PDIFSUMU.B PDIFSUMAU.B	SRX * MULU.W01 * MACCU.W01
1000 1001 1010 1011				
1100 1101 1110 1111				

32	31	28	27	25	20	15	12	7	0
1	f	0	w	rs2	rs1	0 1 0	rd	0 1 1 1 0 1 1	

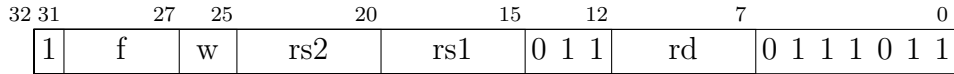
RV32	w			
f	00	01	10	11
000				
001				
010	PSH1ADD.H	*		
011	PSSH1SADD.H	* SSH1SADD		
100				
101				
110	* { UNZIP8P }	*	* { UNZIP8HP }	*
111	* { ZIP8P }	*	* { ZIP8HP }	*

RV64	w			
f	00	01	10	11
000				
001				
010	PSH1ADD.H	* PSH1ADD.W		
011	PSSH1SADD.H	* PSSH1SADD.W		
100				
101				
110	* UNZIP8P	* UNZIP16P	* UNZIP8HP	* UNZIP16HP
111	* ZIP8P	* ZIP16P	* ZIP8HP	* ZIP16HP

32	31	28	27	25	20	15	12	7	0
1	f	1	w	rs2	rs1	0 1 0	rd	0 1 1 1 0 1 1	

RV32	w			
f	00	01	10	11
000	{ PSSL.H }	*	{ PSSL.B }	
001				
010	{ PSSHL.H }	*	{ PSSHL.B }	
011	{ PSSHLR.H }	*	{ PSSHLR.B }	
100				
101				
110	{ PSSHA.H }	*	{ PSSHA.B }	
111	{ PSSHAR.H }	*	{ PSSHAR.B }	

RV64	w			
f	00	01	10	11
000	{ PSSL.H }	* { PSSL.W }	{ PSSL.B }	
001				
010	{ PSSHL.H }	* { PSSHL.W }	{ PSSHL.B }	
011	{ PSSHLR.H }	* { PSSHLR.W }	{ PSSHLR.B }	
100				
101				
110	{ PSSHA.H }	* { PSSHA.W }	{ PSSHA.B }	
111	{ PSSHAR.H }	* { PSSHAR.W }	{ PSSHAR.B }	



RV32 f	w			
	00	01	10	11
0000	PMUL.H.B00	* MUL.H00		*
0001	{ PMACC.H.B00 }	* MACC.H00		*
0010	PMUL.H.B11	* MUL.H11		*
0011	{ PMACC.H.B11 }	* MACC.H11		*
0100	PMULU.H.B00	* MULU.H00		*
0101	{ PMACCU.H.B00 }	* MACCU.H00		*
0110	PMULU.H.B11	* MULU.H11		*
0111	{ PMACCU.H.B11 }	* MACCU.H11		*
1000				
1001				
1010				
1011				
1100	PMULSU.H.B00	* MULSU.H00		*
1101	{ PMACCSU.H.B00 }	* MACCSU.H00		*
1110	PMULSU.H.B11	* MULSU.H11		*
1111	{ PMACCSU.H.B11 }	* MACCSU.H11		*
RV64 f	w			
	00	01	10	11
0000	PMUL.H.B00	* PMUL.W.H00		* MUL.W00
0001	{ PMACC.H.B00 }	* PMACC.W.H00		* MACC.W00
0010	PMUL.H.B11	* PMUL.W.H11		* MUL.W11
0011	{ PMACC.H.B11 }	* PMACC.W.H11		* MACC.W11
0100	PMULU.H.B00	* PMULU.W.H00		* MULU.W00
0101	{ PMACCU.H.B00 }	* PMACCU.W.H00		* MACCU.W00
0110	PMULU.H.B11	* PMULU.W.H11		* MULU.W11
0111	{ PMACCU.H.B11 }	* PMACCU.W.H11		* MACCU.W11
1000				
1001				
1010				
1011				
1100	PMULSU.H.B00	* PMULSU.W.H00		* MULSU.W00
1101	{ PMACCSU.H.B00 }	* PMACCSU.W.H00		* MACCSU.W00
1110	PMULSU.H.B11	* PMULSU.W.H11		* MULSU.W11
1111	{ PMACCSU.H.B11 }	* PMACCSU.W.H11		* MACCSU.W11

32	31	28	27	25	20	15	12	7	0
1	f	0	w	rs2	rs1	1 0 0	rd	0 1 1 1 0 1 1	

RV32		w			
f		00	01	10	11
000	PPACK.H		*		
001	PPACKBT.H		* PACKBT		*
010	PPACKTB.H		* PACKTB		*
011	PPACKT.H		* PACKT		*
100					
101					
110					
111					

RV64		w			
f		00	01	10	11
000	PPACK.H		* PPACK.W		
001	PPACKBT.H		* PPACKBT.W		* PACKBT
010	PPACKTB.H		* PPACKTB.W		* PACKTB
011	PPACKT.H		* PPACKT.W		* PACKT
100					
101					
110					
111					

32	31	28	27	25	20	15	12	7	0
1	f	1	w	rs2	rs1	1 0 0	rd	0 1 1 1 0 1 1	

RV32		w			
f		00	01	10	11
000	{ PSRL.H }		*	{ PSRL.B }	
001					
010					
011					
100	{ PSRA.H }		*	{ PSRA.B }	
101					
110					
111					

RV64		w			
f		00	01	10	11
000	{ PSRL.H }		* { PSRL.W }	{ PSRL.B }	
001					
010					
011					
100	{ PSRA.H }		* { PSRA.W }	{ PSRA.B }	
101					
110					
111					

32	31	27	25	20	15	12	7	0
1	f	w	rs2	rs1	1 0 1	rd	0 1 1 1 0 1 1	

RV32	w			
f	00	01	10	11
0000	PM2ADD.H	*	PM4ADD.B	*
0001	PM2ADDA.H	*	PM4ADDA.B	*
0010	PM2ADD.HX	*		
0011	PM2ADDA.HX	*		
0100	PM2ADDU.H	*	PM4ADDU.B	*
0101	PM2ADDAU.H	*	PM4ADDAU.B	*
0110	PMQ2ADD.H	*	PMQR2ADD.H	*
0111	PMQ2ADDA.H	*	PMQR2ADDA.H	*
1000	PM2SUB.H	*	PM2SADD.H	
1001	PM2SUBA.H	*		
1010	PM2SUB.HX	*	PM2SADD.HX	
1011	PM2SUBA.HX	*		
1100	PM2ADDSU.H	*	PM4ADDSU.B	*
1101	PM2ADDASU.H	*	PM4ADDASU.B	*
1110				
1111	* MQACC.H01	*	* MQRACC.H01	*

RV64	w			
f	00	01	10	11
0000	PM2ADD.H	* PM2ADD.W	PM4ADD.B	* PM4ADD.H
0001	PM2ADDA.H	* PM2ADDA.W	PM4ADDA.B	* PM4ADDA.H
0010	PM2ADD.HX	* PM2ADD.WX		
0011	PM2ADDA.HX	* PM2ADDA.WX		
0100	PM2ADDU.H	* PM2ADDU.W	PM4ADDU.B	* PM4ADDU.H
0101	PM2ADDAU.H	* PM2ADDAU.W	PM4ADDAU.B	* PM4ADDAU.H
0110	PMQ2ADD.H	* PMQ2ADD.W	PMQR2ADD.H	* PMQR2ADD.W
0111	PMQ2ADDA.H	* PMQ2ADDA.W	PMQR2ADDA.H	* PMQR2ADDA.W
1000	PM2SUB.H	* PM2SUB.W	PM2SADD.H	
1001	PM2SUBA.H	* PM2SUBA.W		
1010	PM2SUB.HX	* PM2SUB.WX	PM2SADD.HX	
1011	PM2SUBA.HX	* PM2SUBA.WX		
1100	PM2ADDSU.H	* PM2ADDSU.W	PM4ADDSU.B	* PM4ADDSU.H
1101	PM2ADDASU.H	* PM2ADDASU.W	PM4ADDASU.B	* PM4ADDASU.H
1110				
1111	* PMQACC.W.H01	* MQACC.W01	* PMQRACC.W.H01	* MQRACC.W01

32	31	27	25	20	15	12	7	0
1	f	w	rs2	rs1	1 1 0	rd	0 1 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
0000	PAS.HX	*	PSA.HX	*
0001				
0010	PSAS.HX	*	PSSA.HX	*
0011	PAAS.HX	*	PASA.HX	*
0100				
0101				
0110				
0111				
1000	PMSEQ.H	* MSEQ	PMSEQ.B	
1001				
1010	PMSLT.H	* MSLT	PMSLT.B	
1011	PMSLTU.H	* MSLTU	PMSLTU.B	
1100	PMIN.H	*	PMIN.B	
1101	PMINU.H	*	PMINU.B	
1110	PMAX.H	*	PMAX.B	
1111	PMAXU.H	*	PMAXU.B	
RV64 f	w			
	00	01	10	11
0000	PAS.HX	* PAS.WX	PSA.HX	* PSA.WX
0001				
0010	PSAS.HX	* PSAS.WX	PSSA.HX	* PSSA.WX
0011	PAAS.HX	* PAAS.WX	PASA.HX	* PASA.WX
0100				
0101				
0110				
0111				
1000	PMSEQ.H	* PMSEQ.W	PMSEQ.B	
1001				
1010	PMSLT.H	* PMSLT.W	PMSLT.B	
1011	PMSLTU.H	* PMSLTU.W	PMSLTU.B	
1100	PMIN.H	* PMIN.W	PMIN.B	
1101	PMINU.H	* PMINU.W	PMINU.B	
1110	PMAX.H	* PMAX.W	PMAX.B	
1111	PMAXU.H	* PMAXU.W	PMAXU.B	

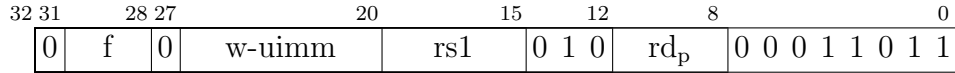
32	31	27	25	20	15	12	7	0							
1	f	w	rs2	rs1	1	1	1	rd	0	1	1	1	0	1	1

RV32	w			
f	00	01	10	11
0000	PMULH.H	*	PMULHR.H	* MULHR
0001	PMHACC.H	* MHACC	PMHRACC.H	* MHRACC
0010	PMULHU.H	*	PMULHRU.H	* MULHRU
0011	PMHACCU.H	* MHACCU	PMHRACCU.H	* MHRACCU
0100	PMULH.H.B0	* MULH.H0	PMULHSU.H.B0	* MULHSU.H0
0101	PMHACC.H.B0	* MHACC.H0	PMHACCSU.H.B0	* MHACCSU.H0
0110	PMULH.H.B1	* MULH.H1	PMULHSU.H.B1	* MULHSU.H1
0111	PMHACC.H.B1	* MHACC.H1	PMHACCSU.H.B1	* MHACCSU.H1
1000	PMULHSU.H	*	PMULHRSU.H	* MULHRSU
1001	PMHACCSU.H	* MHACCSU	PMHRACCSU.H	* MHRACCSU
1010	PMULQ.H	* MULQ	PMULQR.H	* MULQR
1011				
1100				
1101	* MQACC.H00	*	* MQRACC.H00	*
1110				
1111	* MQACC.H11	*	* MQRACC.H11	*

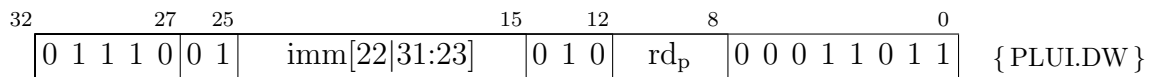
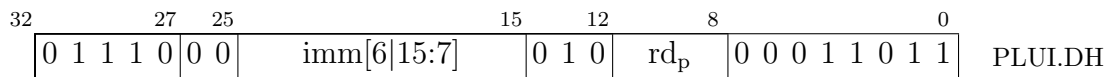
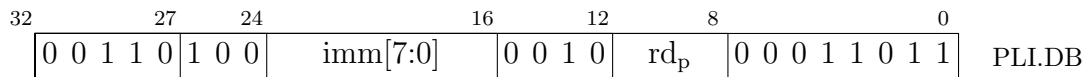
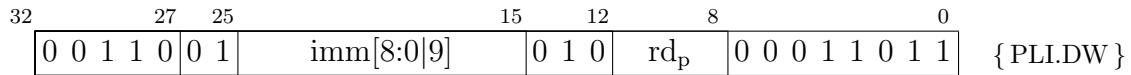
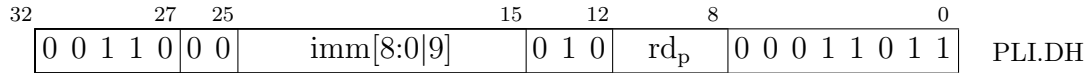
RV64	w			
f	00	01	10	11
0000	PMULH.H	* PMULH.W	PMULHR.H	* PMULHR.W
0001	PMHACC.H	* PMHACC.W	PMHRACC.H	* PMHRACC.W
0010	PMULHU.H	* PMULHU.W	PMULHRU.H	* PMULHRU.W
0011	PMHACCU.H	* PMHACCU.W	PMHRACCU.H	* PMHRACCU.W
0100	PMULH.H.B0	* PMULH.W.H0	PMULHSU.H.B0	* PMULHSU.W.H0
0101	PMHACC.H.B0	* PMHACC.W.H0	PMHACCSU.H.B0	* PMHACCSU.W.H0
0110	PMULH.H.B1	* PMULH.W.H1	PMULHSU.H.B1	* PMULHSU.W.H1
0111	PMHACC.H.B1	* PMHACC.W.H1	PMHACCSU.H.B1	* PMHACCSU.W.H1
1000	PMULHSU.H	* PMULHSU.W	PMULHRSU.H	* PMULHRSU.W
1001	PMHACCSU.H	* PMHACCSU.W	PMHRACCSU.H	* PMHRACCSU.W
1010	PMULQ.H	* PMULQ.W	PMULQR.H	* PMULQR.W
1011				
1100				
1101	* PMQACC.W.H00	* MQACC.W00	* PMQRACC.W.H00	* MQRACC.W00
1110				
1111	* PMQACC.W.H11	* MQACC.W11	* PMQRACC.W.H11	* MQRACC.W11

3 Encodings of instructions with one or more register-pair operands (RV32 only)

As before, where an instruction name is within braces { }, there is no such instruction proposed for the Base P extension, but the indicated encoding is recommended should the instruction be provided by another extension.



RV32	w-uimm	000xxxx	001xxxx	01xxxxx	1xxxxxx
f					
000			PWSLLI.B	PWSLLI.H	WSLLI
001					
010					
011	<i>PLI instructions</i>				
100			PWSLAI.B	PWSLAI.H	WSLAI
101					
110					
111	<i>PLUI instructions</i>				

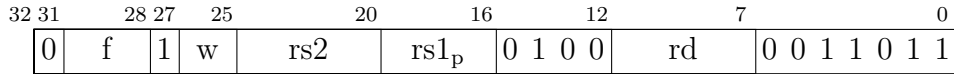


32	31	28	27	25	20	15	12	8	0
0	f	1	w	rs2	rs1	0 1 0	rd _p	0 0 0 1 1 0 1 1	

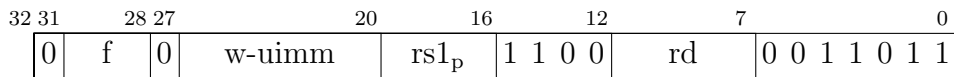
RV32 f	w			
	00	01	10	11
000 001 010 011	PWSSL.BS	PWSSL.HS		WSSL
100 101 110 111	PWSLA.BS WZIP8P	PWSLA.HS WZIP16P		WSLA

32	31	27	25	20	15	12	8	0
0	f	w	rs2	rs1	0 1 0	rd _p	1 0 0 1 1 0 1 1	

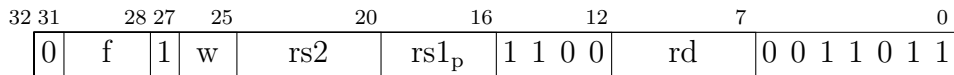
RV32 f	w			
	00	01	10	11
0000 0001 0010 0011	PWADD.H PWADDA.H PWADDU.H PWADDAU.H	WADD WADDA WADDU WADDAU	PWADD.B PWADDA.B PWADDU.B PWADDAU.B	PM2WADD.H PM2WADDA.H PM2WADD.HX PM2WADDA.HX
0100 0101 0110 0111	PWMUL.H PWMACC.H PWMULU.H PWMACCU.H	WMUL WMACC WMULU WMACCU	PWMUL.B { PWMACC.B } PWMULU.B { PWMACCU.B }	PM2WADDU.H PM2WADDAU.H
1000 1001 1010 1011	PWSUB.H PWSUBA.H PWSUBU.H PWSUBAU.H	WSUB WSUBA WSUBU WSUBAU	PWSUB.B PWSUBA.B PWSUBU.B PWSUBAU.B	PM2WSUB.H PM2WSUBA.H PM2WSUB.HX PM2WSUBA.HX
1100 1101 1110 1111	PWMULSU.H PWMACCSU.H PMQWACC.H	WMULSU WMACCSU PMQWACC	PWMULSU.B { PWMACCSU.B } PMQRWACC.H	PM2WADDSU.H PM2WADDASU.H PMQRWACC



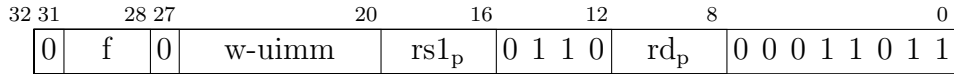
RV32 f	w	00	01	10	11
000					
001		PREDSUM.DHS		PREDSUM.DBS	
010					
011		PREDSUMU.DHS		PREDSUMU.DBS	
100					
101					
110					
111					



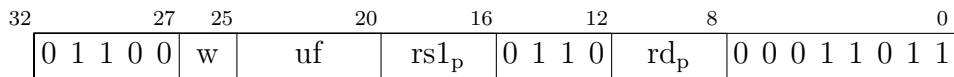
RV32 f	w-uimm	000xxxx	001xxxx	01xxxxx	1xxxxxx
000			PNSRLI.B	PNSRLI.H	NSRLI
001			{ PNSRLRI.B }	{ PNSRLRI.H }	{ NSRLRI }
010			PNCLIPIU.B	PNCLIPIU.H	NCLIPIU
011			PNCLIPRIU.B	PNCLIPRIU.H	NCLIPRIU
100			PNSRAI.B	PNSRAI.H	NSRAI
101			PNSRARI.B	PNSRARI.H	NSRARI
110			PNCLIP.I.B	PNCLIP.I.H	NCLIP.I
111			PNCLIPRI.B	PNCLIPRI.H	NCLIPRI



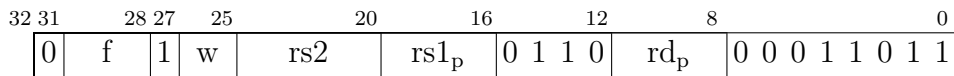
RV32 f	w	00	01	10	11
000		PNSRL.BS	PNSRL.HS		NSRL
001		{ PNSRLR.BS }	{ PNSRLR.HS }		{ NSRLR }
010		PNCLIPU.BS	PNCLIPU.HS		NCLIPU
011		PNCLIPRU.BS	PNCLIPRU.HS		NCLIPRU
100		PNSRA.BS	PNSRA.HS		NSRA
101		PNSRAR.BS	PNSRAR.HS		NSRAR
110		PNCLIP.BS	PNCLIP.HS		NCLIP
111		PNCLIPR.BS	PNCLIPR.HS		NCLIPR



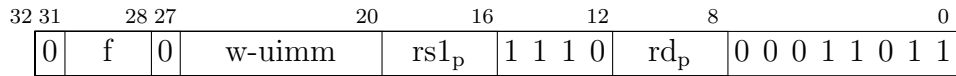
RV32 f	w-uimm				
	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSLLI.DB	PSLLI.DH	PSLLI.DW	
001		{ PSSLI.DB }	{ PSSLI.DH }	{ PSSLI.DW }	
010					
011					
100					
101		{ PSSLAI.DB }	PSSLAI.DH	PSSLAI.DW	
110	<i>unary instructions</i>				
111					



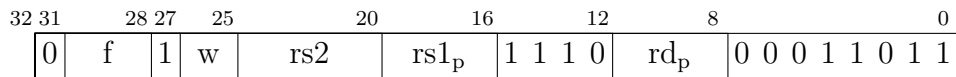
RV32 uf	w			
	00	01	10	11
00000	{ PCLZ.DH }	{ PCLZ.DW }	{ PCLZ.DB }	
00001				
00010				
00011	{ PCLS.DH }	{ PCLS.DW }	{ PCLS.DB }	
00100	PSEXT.DH.B	PSEXT.DW.B		
00101		PSEXT.DW.H		
00110				
00111	PSABS.DH	{ PSABS.DW }	PSABS.DB	
01000				
01001				
01010				
01011				
...				
11111				



RV32 f	w			
	00	01	10	11
000	PSLL.DHS	PSLL.DWS	PSLL.DBS	
001	PADD.DHS	PADD.DWS	PADD.DBS	
010	{ PSSHL.DHS }	{ PSSHL.DWS }	{ PSSHL.DBS }	
011	{ PSSHLR.DHS }	{ PSSHLR.DWS }	{ PSSHLR.DBS }	
100				
101				
110	PSSHA.DHS	PSSHA.DWS	{ PSSHA.DBS }	
111	PSSHAR.DHS	PSSHAR.DWS	{ PSSHAR.DBS }	



RV32 f	w-uimm				
	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSRLI.DB	PSRLI.DH	PSRLI.DW	
001		{ PSRLRI.DB }	{ PSRLRI.DH }	{ PSRLRI.DW }	
010		{ PUSATI.DB }	PUSATI.DH	PUSATI.DW	
011					
100		PSRAI.DB	PSRAI.DH	PSRAI.DW	
101		{ PSRARI.DB }	PSRARI.DH	PSRARI.DW	
110		{ PSATI.DB }	PSATI.DH	PSATI.DW	
111					



RV32 f	w			
	00	01	10	11
000	PSRL.DHS	PSRL.DWS	PSRL.DBS	
001				
010				
011				
100	PSRA.DHS	PSRA.DWS	PSRA.DBS	
101				
110				
111				

32	31	27	25	21	20	16	12	8	0
1	f	w	rs2 _p	0	rs1 _p	0 1 1 0	rd _p	0 0 0 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
0000	PADD.DH	PADD.DW	PADD.DB	ADDD
0001				
0010	PSADD.DH	PSADD.DW	PSADD.DB	
0011	PAADD.DH	PAADD.DW	PAADD.DB	
0100				
0101				
0110	PSADDU.DH	PSADDU.DW	PSADDU.DB	
0111	PAADDU.DH	PAADDU.DW	PAADDU.DB	
1000	PSUB.DH	PSUB.DW	PSUB.DB	SUBD
1001	PDIF.DH	{ PDIF.DW }	PDIF.DB	
1010	PSSUB.DH	PSSUB.DW	PSSUB.DB	
1011	PASUB.DH	PASUB.DW	PASUB.DB	
1100				
1101	PDIFU.DH	{ PDIFU.DW }	PDIFU.DB	
1110	PSSUBU.DH	PSSUBU.DW	PSSUBU.DB	
1111	PASUBU.DH	PASUBU.DW	PASUBU.DB	

32	31	28	27	25	21	20	16	12	8	0
1	f	0	w	rs2 _p	1	rs1 _p	0 1 1 0	rd _p	0 0 0 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
000				
001				
010	PSH1ADD.DH	PSH1ADD.DW		
011	PSSH1SADD.DH	PSSH1SADD.DW		
100				
101				
110				
111				

32	31	28	27	25	21	20	16	12	8	0
1	f	1	w	rs2 _p	1	rs1 _p	0 1 1 0	rd _p	0 0 0 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
000	{ PSL.L.DH }	{ PSL.L.DW }	{ PSL.L.DB }	
001				
010	{ PSSHL.DH }	{ PSSHL.DW }	{ PSSHL.DB }	
011	{ PSSHLR.DH }	{ PSSHLR.DW }	{ PSSHLR.DB }	
100				
101				
110	{ PSSHA.DH }	{ PSSHA.DW }	{ PSSHA.DB }	
111	{ PSSHAR.DH }	{ PSSHAR.DW }	{ PSSHAR.DB }	

32	31	28	27	25	21	20	16	12	8	0
1	f	0	w	rs _{2p}	0	rs _{1p}	1 1 1 0	rd _p	0 0 0 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
000	PPACK.DH	PPACK.DW		
001	PPACKBT.DH	PPACKBT.DW		
010	PPACKTB.DH	PPACKTB.DW		
011	PPACKT.DH	PPACKT.DW		
100				
101				
110				
111				

32	31	28	27	25	21	20	16	12	8	0
1	f	1	w	rs _{2p}	0	rs _{1p}	1 1 1 0	rd _p	0 0 0 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
000	{PSRL.DH}	{PSRL.DW}	{PSRL.DB}	
001				
010				
011				
100	{PSRA.DH}	{PSRA.DW}	{PSRA.DB}	
101				
110				
111				

32	31	27	25	21	20	16	12	8	0
1	f	w	rs _{2p}	1	rs _{1p}	1 1 1 0	rd _p	0 0 0 1 1 0 1 1	

RV32 f	w			
	00	01	10	11
0000	PAS.DHX		PSA.DHX	
0001				
0010	PSAS.DHX		PSSA.DHX	
0011	PAAS.DHX		PASA.DHX	
0100				
0101				
0110				
0111				
1000	PMSEQ.DH	PMSEQ.DW	PMSEQ.DB	
1001				
1010	PMSLT.DH	PMSLT.DW	PMSLT.DB	
1011	PMSLTU.DH	PMSLTU.DW	PMSLTU.DB	
1100	PMIN.DH		PMIN.DB	
1101	PMINU.DH		PMINU.DB	
1110	PMAX.DH		PMAX.DB	
1111	PMAXU.DH		PMAXU.DB	