Proposed Instruction Encodings for the RISC-V Base P Extension

John Hauser

July 1, 2024

Warning! This document is only a draft proposal. The Base P extension that is eventually ratified by the RISC-V International Association 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 extension, see these other documents:

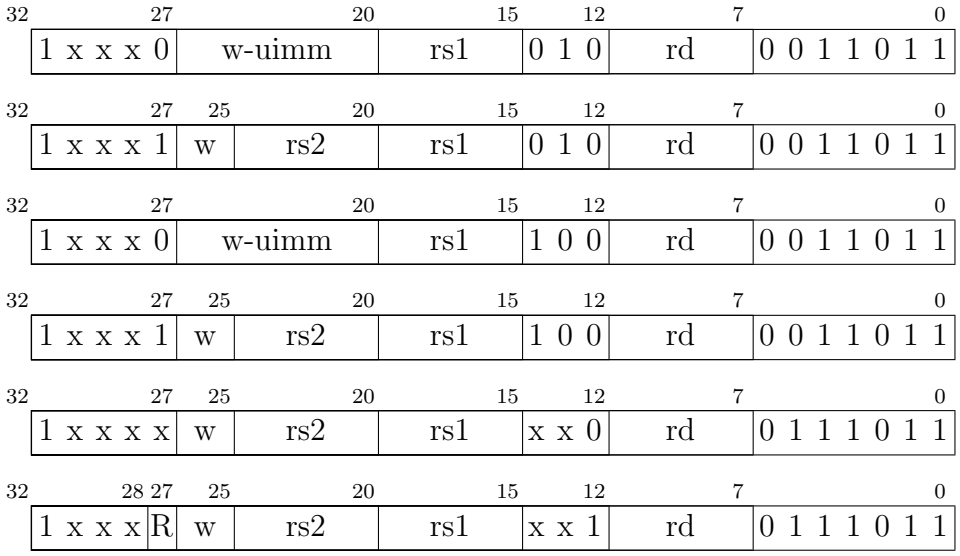
- *System for RISC-V P Extension Instruction Names*
- *Proposed Instructions for the RISC-V Base P Extension*

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 almost all Base P instructions that have no register-pair operands, only single registers:

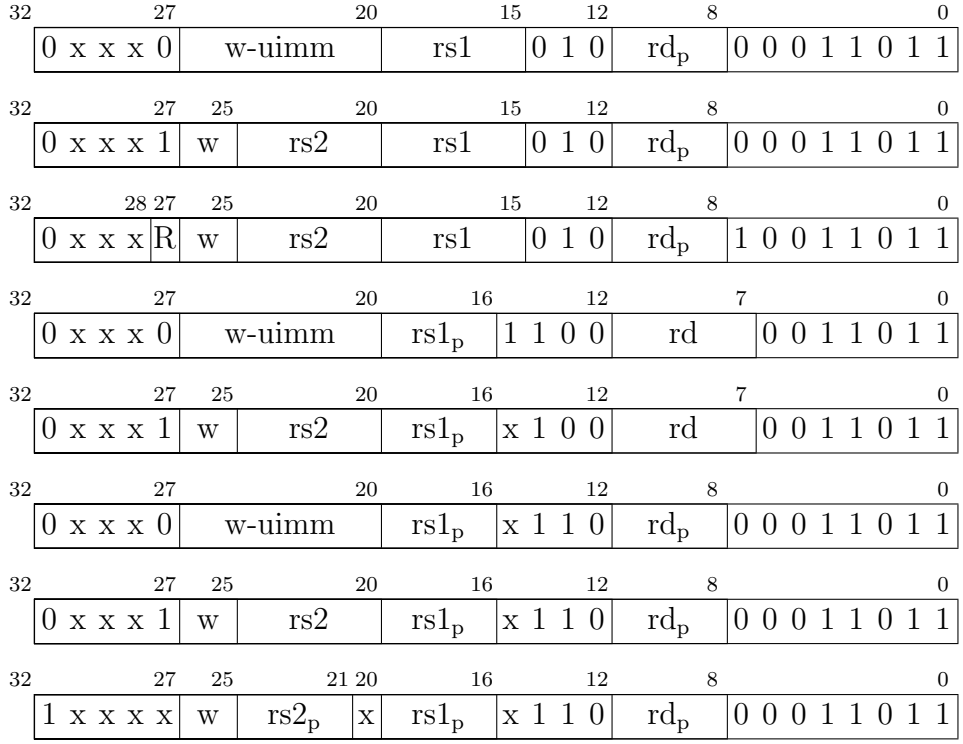


Note that the first four formats are in major opcode OP-IMM-32, and the last two are in OP-32. Of the OP-IMM-32 formats, those with func3 = binary 010 are mostly shifts left, while those with func3 = binary 100 are mostly shifts right.

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.

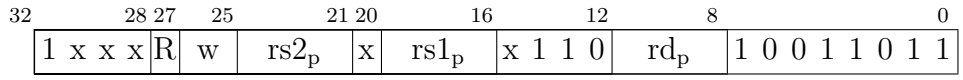
With RV32, the P extension has additional instruction formats for the cases when operands are even-odd register pairs:



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.).

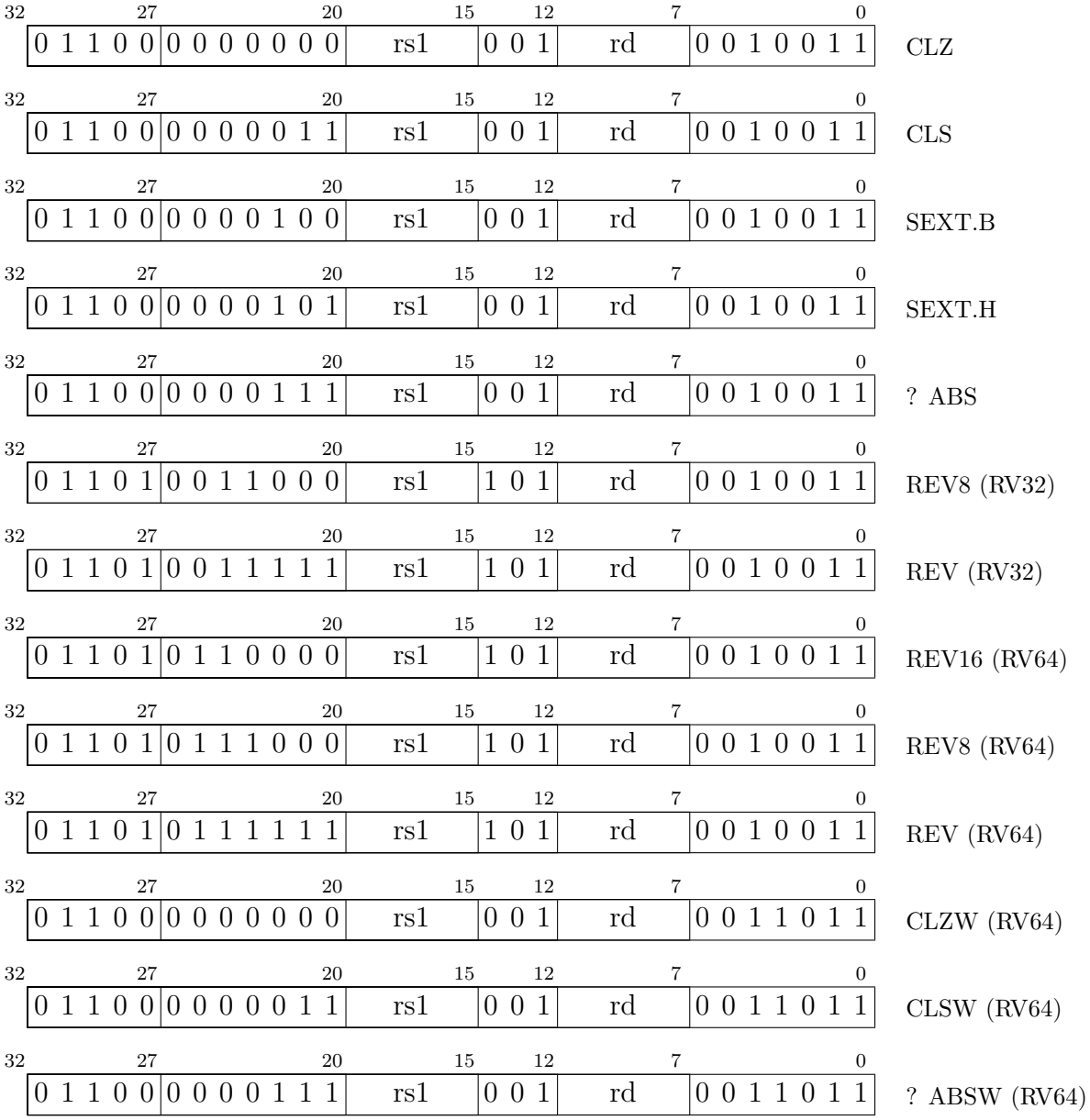
The following format is not used by Base P but is recommended for a Zp∗ extension that adds double-wide multiply instructions:

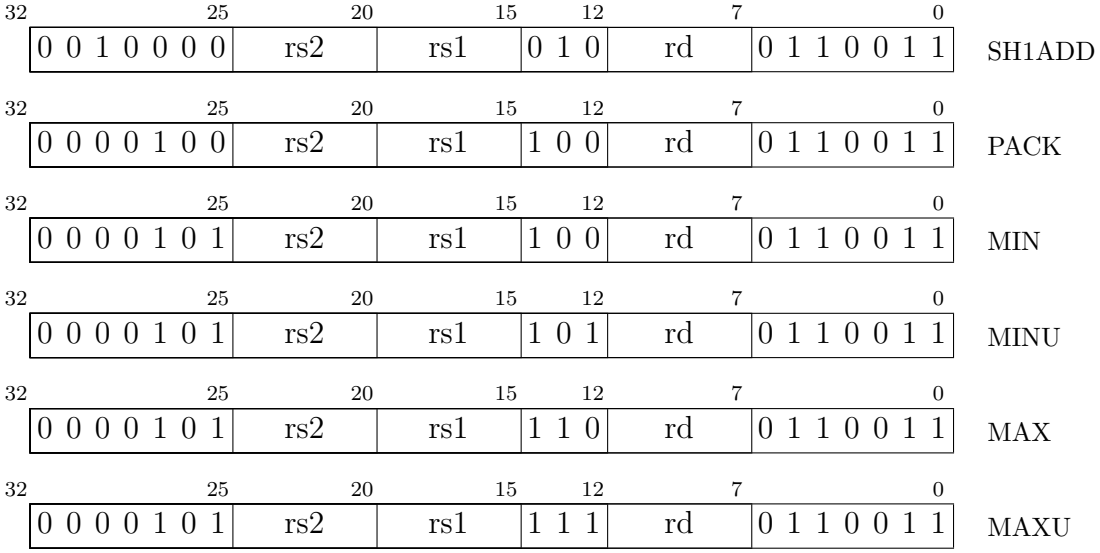


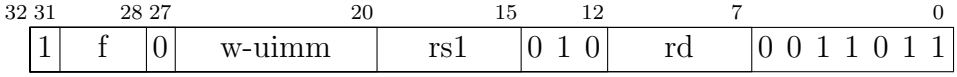
2 Encodings of instructions with no register-pair operands

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 in the Base P extension, but the indicated encoding is recommended should the instruction be provided by another extension.

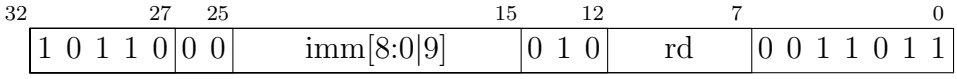



32 25 20 15 12 7 0
0 0 1 0 0 0 0
rs2
rs1
0 1 0
rd
0 1 1 0 0 1 1
SH1ADD
32 25 20 15 12 7 0
0 0 0 0 1 0 0
rs2
rs1
1 0 0
rd
0 1 1 0 0 1 1
PACK
32 25 20 15 12 7 0
0 0 0 0 1 0 1
rs2
rs1
1 0 0
rd
0 1 1 0 0 1 1
MIN
32 25 20 15 12 7 0
0 0 0 0 1 0 1
rs2
rs1
1 0 1
rd
0 1 1 0 0 1 1
MINU
32 25 20 15 12 7 0
0 0 0 0 1 0 1
rs2
rs1
1 1 0
rd
0 1 1 0 0 1 1
MAX
32 25 20 15 12 7 0
0 0 0 0 1 0 1
rs2
rs1
1 1 1
rd
0 1 1 0 0 1 1
MAXU

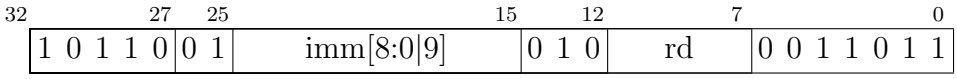


RV32 f	w-uimm 0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSLLI.B { PSSLLI.B }	PSLLI.H { PSSLLI.H }	* * { SSLLI }	
001					
010					
011	*PLI instructions*				
100					
101		{ PSSLAI.B }	PSSLAI.H	* SSLAI	
110	*unary instructions*				
111	*PLUI instructions*				

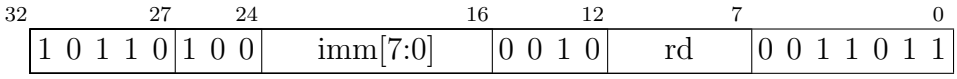
RV64 f	w-uimm 0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSLLI.B { PSSLLI.B }	PSLLI.H { PSSLLI.H }	* PSLLI.W * { PSSLLI.W }	
001					
010					
011	*PLI instructions*				
100					
101		{ PSSLAI.B }	PSSLAI.H	* PSSLAI.W	
110	*unary instructions*				
111	*PLUI instructions*				



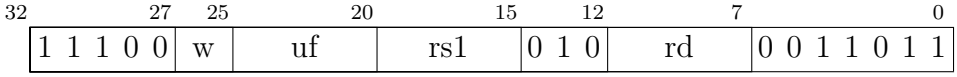
PLI.H



PLI.W (RV64)

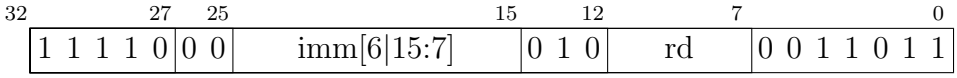


PLI.B

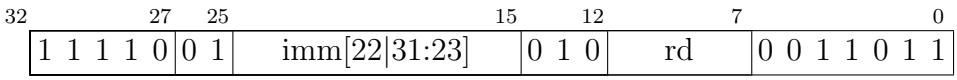


RV32	w			
uf	00	01	10	11
00000	{PCLZ.H}	*	{PCLZ.B}	
00001				
00010				
00011	{PCLS.H}	*	{PCLS.B}	
00100	PSEXTB.H	*		
00101		*		
00110				
00111	PSABS.H	*{SABS}	PSABS.B	
01000				
01001				
01010				
01011				
...				
11111				

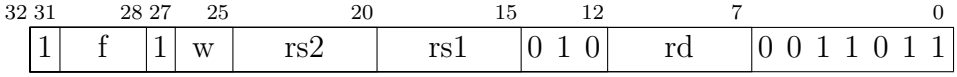
RV64	w			
uf	00	01	10	11
00000	{PCLZ.H}	*{PCLZ.W}	{PCLZ.B}	
00001				
00010				
00011	{PCLS.H}	*{PCLS.W}	{PCLS.B}	
00100	PSEXTB.H	* PSEXTB.W		
00101		* PSEXTH.W		
00110				
00111	PSABS.H	*{PSABS.W}	PSABS.B	
01000				
01001				
01010				
01011				
...				
11111				



PLUI.H

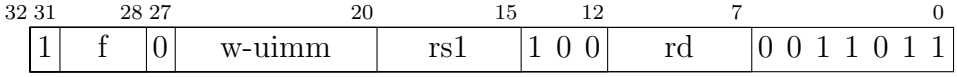


PLUI.W (RV64)



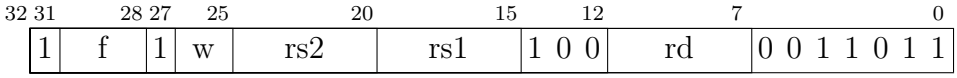
RV32	w			
f	00	01	10	11
000	PSLL.H.H0	*	PSLL.B.B0	
001	PADD.H.H0	*	PADD.B.B0	
010	{ PSSHL.H.H0 }	* { SSHL }	{ PSSHL.B.B0 }	*
011	{ PSSHLR.H.H0 }	* { SSHLR }	{ PSSHLR.B.B0 }	*
100				
101				
110	PSSHA.H.H0	* SSHA	{ PSSHA.B.B0 }	*
111	PSSHAR.H.H0	* SSHAR	{ PSSHAR.B.B0 }	*

RV64	w			
f	00	01	10	11
000	PSLL.H.H0	* PSLL.W.W0	PSLL.B.B0	
001	PADD.H.H0	* PADD.W.W0	PADD.B.B0	
010	{ PSSHL.H.H0 }	* { PSSHL.W.W0 }	{ PSSHL.B.B0 }	* { SHL }
011	{ PSSHLR.H.H0 }	* { PSSHLR.W.W0 }	{ PSSHLR.B.B0 }	* { SHLR }
100				
101				
110	PSSHA.H.H0	* PSSHA.W.W0	{ PSSHA.B.B0 }	* SHA
111	PSSHAR.H.H0	* PSSHAR.W.W0	{ PSSHAR.B.B0 }	* SHAR



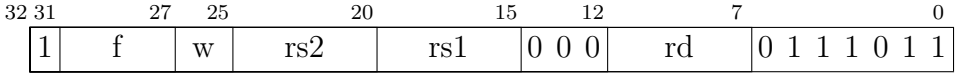
RV32	w-uimm				
f	0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSRLI.B	PSRLI.H	*	
001		{ PSRLRI.B }	{ PSRLRI.H }	* { SRLRI }	*
010		{ PUSATI.B }	PUSATI.H	* USATI	*
011					
100		PSRAI.B	PSRAI.H	*	
101		{ PSRARI.B }	PSRARI.H	* SRARI	*
110		{ PSATI.B }	PSATI.H	* SATI	*
111					

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					



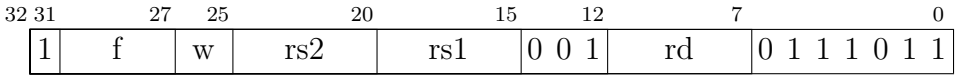
RV32	w			
f	00	01	10	11
000	PSRL.H.H0	*	PSRL.B.B0	
001	PREDSUM.H	*	? PREDSUM.B	
010				
011	PREDSUMU.H	*	? PREDSUMU.B	
100	PSRA.H.H0	*	PSRA.B.B0	
101				
110				
111				

RV64	w			
f	00	01	10	11
000	PSRL.H.H0	* PSRL.W.W0	PSRL.B.B0	
001	PREDSUM.H	* PREDSUM.W	? PREDSUM.B	
010				
011	PREDSUMU.H	* PREDSUMU.W	? PREDSUMU.B	
100	PSRA.H.H0	* PSRA.W.W0	PSRA.B.B0	
101				
110				
111				



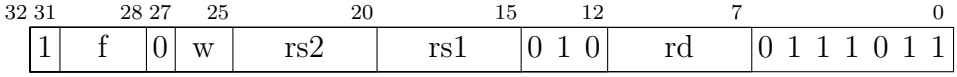
RV32	w			
f	00	01	10	11
0000	PADD.H	*	PADD.B	
0001				
0010	PSADD.H	* SADD	PSADD.B	
0011	PAADD.H	* AADD	PAADD.B	
0100				
0101				
0110	PSADDU.H	* SADDU	PSADDU.B	
0111	PAADDU.H	* AADDU	PAADDU.B	
1000	PSUB.H	*	PSUB.B	
1001	PDIF.H	* { DIF }	PDIF.B	
1010	PSSUB.H	* SSUB	PSSUB.B	
1011	PASUB.H	* ASUB	PASUB.B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

RV64	w			
f	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	



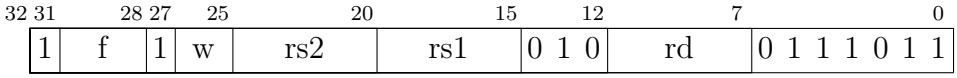
RV32 f	w 00	01	10	11
0000				? SLX
0001				*
0010	PMUL.H.BEO	* MUL.H01		*
0011	{ PMACC.H.BEO }	* MACC.H01		*
0100				
0101	MVM	MVMN	MERGE	SRX
0110	PMULU.H.BEO	* MULU.H01	PDIFSUMU.B	*
0111	{ PMACCU.H.BEO }	* MACCU.H01	PDIFSUMAU.B	*
1000				
1001				
1010				
1011				
1100				
1101				
1110				
1111				

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



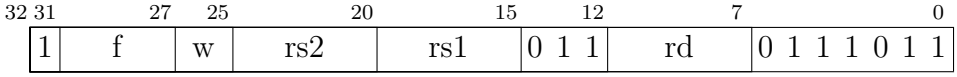
RV32	w			
f	00	01	10	11
000				
001				
010	PSH1ADD.H	*		
011	PSSH1SADD.H	* SSH1SADD		
100				
101				
110	* { UNZIP8P }	*	* { UNZIP8HP }	*
111	* { ZIP8P }	*	* { ZIP8HP }	*

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



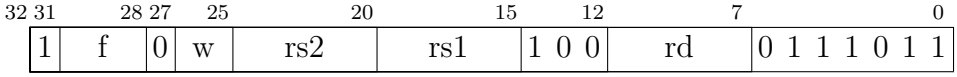
RV32	w			
f	00	01	10	11
000	{ PSLL.H }	*	{ PSLL.B }	
001				
010	{ PSSHL.H }	*	{ PSSHL.B }	
011	{ PSSHLR.H }	*	{ PSSHLR.B }	
100				
101				
110	{ PSSHA.H }	*	{ PSSHA.B }	
111	{ PSSHAR.H }	*	{ PSSHAR.B }	

RV64	w			
f	00	01	10	11
000	{ PSLL.H }	* { PSLL.W }	{ PSLL.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 }	



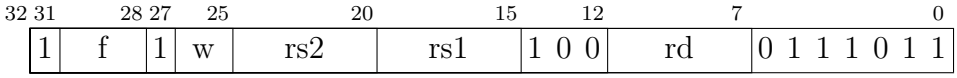
RV32 f	w 00	01	10	11
0000	PMUL.H.BEE	* MUL.H00		*
0001	{ PMACC.H.BEE }	* MACC.H00		*
0010	PMUL.H.BOO	* MUL.H11		*
0011	{ PMACC.H.BOO }	* MACC.H11		*
0100	PMULU.H.BEE	* MULU.H00		*
0101	{ PMACCU.H.BEE }	* MACCU.H00		*
0110	PMULU.H.BOO	* MULU.H11		*
0111	{ PMACCU.H.BOO }	* MACCU.H11		*
1000				
1001		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

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



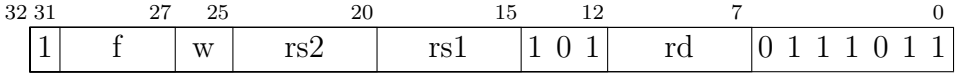
RV32	w			
f	00	01	10	11
000	PPACK.H	*		
001	PPACKBT.H	* PACKBT		*
010	PPACKTB.H	* PACKTB		*
011	PPACKT.H	* PACKT		*
100				
101				
110				
111				

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				



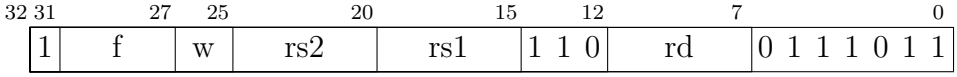
RV32	w			
f	00	01	10	11
000	{ PSRL.H }	*	{ PSRL.B }	
001				
010				
011				
100	{ PSRA.H }	*	{ PSRA.B }	
101				
110				
111				

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				



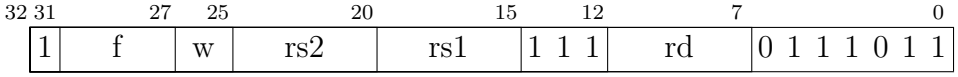
RV32 f	w 00	01	10	11
0000	PM2ADD.H	*	PM4ADD.B	*
0001	PM2ADDA.H	*	PM4ADDA.B	*
0010	PM2ADD.HX	*		
0011	PM2ADDA.HX	*		
0100	PM2ADDU.H	*	PM4ADDU.B	*
0101	PM2ADDAU.H	*	PM4ADDAU.B	*
0110	PMQ2ADD.H	*	PMQR2ADD.H	*
0111	PMQ2ADDA.H	*	PMQR2ADDA.H	*
1000	PM2SUB.H	*	PM2SADD.H	
1001	PM2SUBA.H	*		
1010	PM2SUB.HX	*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

RV64 f	w 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.HEO	* MQACC.W01	* PMQRACC.W.HEO	* MQRACC.W01



RV32	w			
f	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	w			
f	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	

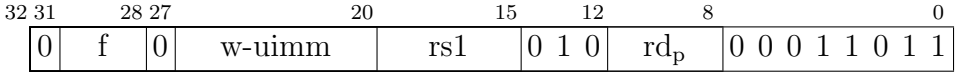


RV32	w			
f	00	01	10	11
0000	PMULH.H	*	PMULHR.H	* MULHR
0001	PMHACC.H	* MHACC	PMHRACC.H	* MHRACC
0010	PMULHU.H	*	PMULHRU.H	* MULHRU
0011	PMHACCU.H	* { MHACCU }	PMHRACCU.H	* { MHRACCU }
0100	PMULH.H.BE	* MULH.H0	PMULHSU.H.BE	* MULHSU.H0
0101	PMHACC.H.BE	* MHACC.H0	PMHACCSU.H.BE	* MHACCSU.H0
0110	PMULH.H.BO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

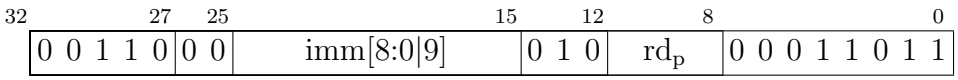
RV64	w			
f	00	01	10	11
0000	PMULH.H	* PMULH.W	PMULHR.H	* PMULHR.W
0001	PMHACC.H	* PMHACC.W	PMHRACC.H	* PMHRACC.W
0010	PMULHU.H	* PMULHU.W	PMULHRU.H	* PMULHRU.W
0011	PMHACCU.H	* { PMHACCU.W }	PMHRACCU.H	* { PMHRACCU.W }
0100	PMULH.H.BE	* PMULH.W.HE	PMULHSU.H.BE	* PMULHSU.W.HE
0101	PMHACC.H.BE	* PMHACC.W.HE	PMHACCSU.H.BE	* PMHACCSU.W.HE
0110	PMULH.H.BO	* PMULH.W.HO	PMULHSU.H.BO	* PMULHSU.W.HO
0111	PMHACC.H.BO	* PMHACC.W.HO	PMHACCSU.H.BO	* PMHACCSU.W.HO
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	PMQACC.H	* PMQACC.W	PMQRACC.H	* PMQRACC.W
1100				
1101	* PMQACC.W.HEE	* MQACC.W00	* PMQRACC.W.HEE	* MQRACC.W00
1110				
1111	* PMQACC.W.HOO	* MQACC.W11	* PMQRACC.W.HOO	* MQRACC.W11

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

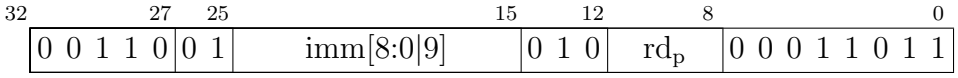
As before, where an instruction name is within braces { }, there is no such instruction in the Base P extension, but the indicated encoding is recommended should the instruction be provided by another extension.



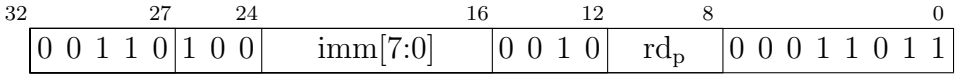
RV32	w-uimm			
	000xxxx	001xxxx	01xxxxx	1xxxxxx
000		PWSLLI.B	PWSLLI.H	WSLLI
001				
010				
011	*PLI instructions*			
100		PWSLAI.B	PWSLAI.H	WSLAI
101				
110				
111	*PLUI instructions*			



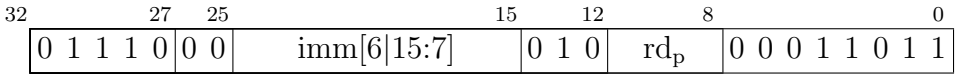
PLI.DH



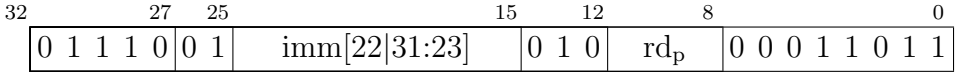
{ PLI.DW }



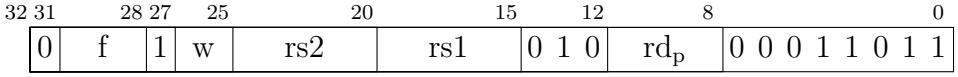
PLI.DB



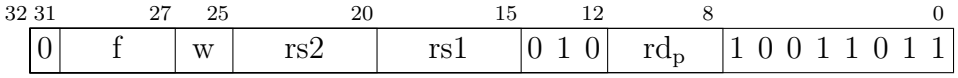
PLUI.DH



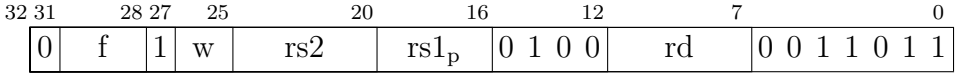
{ PLUI.DW }



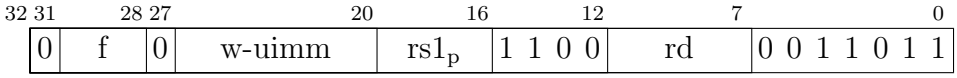
RV32 f	w 00	01	10	11
000	PWSLL.B.B0	PWSLL.H.H0		WSLL
001				
010				
011				
100	PWSLA.B.B0	PWSLA.H.H0		WSLA
101				
110				
111	WZIP8P	WZIP16P		



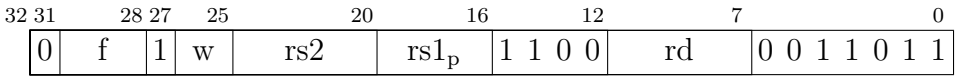
RV32 f	w 00	01	10	11
0000	PWADD.H	WADD	PWADD.B	PM2WADD.H
0001	PWADDA.H	WADDA	PWADDA.B	PM2WADDA.H
0010	PWADDU.H	WADDU	PWADDU.B	PM2WADD.HX
0011	PWADDAU.H	WADDAU	PWADDAU.B	PM2WADDA.HX
0100	PWMUL.H	WMUL	PWMUL.B	PM2WADDU.H
0101	PWMACC.H	WMACC	{ PWMACC.B }	PM2WADDAU.H
0110	PWMULU.H	WMULU	PWMULU.B	
0111	PWMACCU.H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



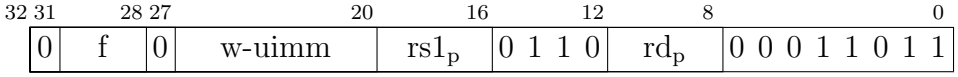
RV32	w			
f	00	01	10	11
000				
001	PREDSUM.DH		? PREDSUM.DB	
010				
011	PREDSUMU.DH		? PREDSUMU.DB	
100				
101				
110				
111				



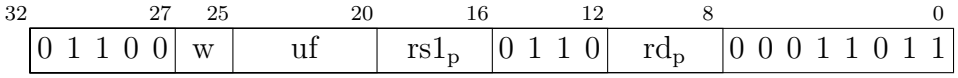
RV32	w-uimm			
	000xxxx	001xxxx	01xxxxx	1xxxxxx
000		PNSRLI.B	PNSRLI.H	NSRLI
001		{PNSRLRI.B}	{PNSRLRI.H}	{NSRLRI}
010		PNUCLIPI.B	PNUCLIPI.H	NUCLIPI
011		PNUCLIPRI.B	PNUCLIPRI.H	NUCLIPRI
100		PNSRAI.B	PNSRAI.H	NSRAI
101		PNSRARI.B	PNSRARI.H	NSRARI
110		PNCLIPI.B	PNCLIPI.H	NCLIPI
111		PNCLIPRI.B	PNCLIPRI.H	NCLIPRI



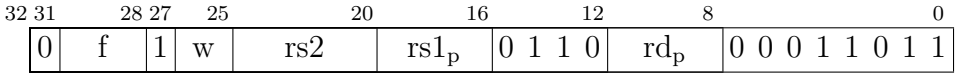
RV32	w			
f	00	01	10	11
000	PNSRL.B.B0	PNSRL.H.H0		NSRL
001	{PNSRLR.B.B0}	{PNSRLR.H.H0}		{NSRLR}
010	PNUCLIP.B.B0	PNUCLIP.H.H0		NUCLIP
011	PNUCLIPR.B.B0	PNUCLIPR.H.H0		NUCLIPR
100	PNSRA.B.B0	PNSRA.H.H0		NSRA
101	PNSRAR.B.B0	PNSRAR.H.H0		NSRAR
110	PNCLIP.B.B0	PNCLIP.H.H0		NCLIP
111	PNCLIPR.B.B0	PNCLIPR.H.H0		NCLIPR



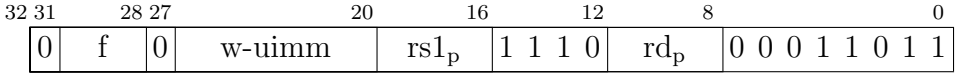
RV32 f	w-uimm 0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSLLI.DB	PSLLI.DH	PSLLI.DW	
001		{ PSSLLI.DB }	{ PSSLLI.DH }	{ PSSLLI.DW }	
010					
011					
100					
101		{ PSSLAI.DB }	PSSLAI.DH	PSSLAI.DW	
110	*unary instructions*				
111					



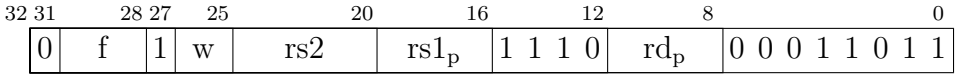
RV32 uf	w 00	01	10	11
00000	{ PCLZ.DH }	{ PCLZ.DW }	{ PCLZ.DB }	
00001				
00010				
00011	{ PCLS.DH }	{ PCLS.DW }	{ PCLS.DB }	
00100	PSEXTB.DH	PSEXTB.DW		
00101		PSEXTH.DW		
00110				
00111	PSABS.DH	{ PSABS.DW }	PSABS.DB	
01000				
01001				
01010				
01011				
...				
11111				



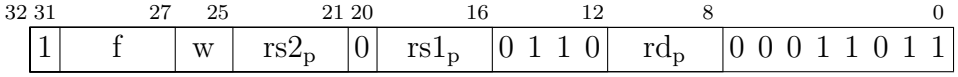
RV32 f	w 00	01	10	11
000	PSLL.DH.H0	PSLL.DW.W	PSLL.DB.B0	
001	PADD.DH.H0	PADD.DW.W	PADD.DB.B0	
010	{ PSSHL.DH.H0 }	{ PSSHL.DW.W }	{ PSSHL.DB.B0 }	
011	{ PSSHLR.DH.H0 }	{ PSSHLR.DW.W }	{ PSSHLR.DB.B0 }	
100				
101				
110	PSSHA.DH.H0	PSSHA.DW.W	{ PSSHA.DB.B0 }	
111	PSSHAR.DH.H0	PSSHAR.DW.W	{ PSSHAR.DB.B0 }	



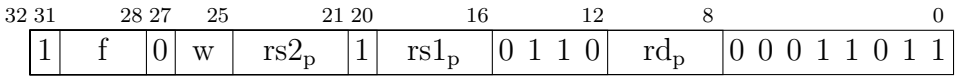
RV32 f	w-uimm 0000xxx	0001xxx	001xxxx	01xxxxx	1xxxxxx
000		PSRLI.DB	PSRLI.DH	PSRLI.DW	
001		{ PSRLRI.DB }	{ PSRLRI.DH }	{ PSRLRI.DW }	
010		{ PUSATI.DB }	PUSATI.DH	PUSATI.DW	
011					
100		PSRAI.DB	PSRAI.DH	PSRAI.DW	
101		{ PSRARI.DB }	PSRARI.DH	PSRARI.DW	
110		{ PSATI.DB }	PSATI.DH	PSATI.DW	
111					



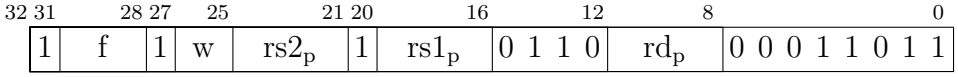
RV32 f	w 00	01	10	11
000	PSRL.DH.H0	PSRL.DW.W	PSRL.DB.B0	
001				
010				
011				
100	PSRA.DH.H0	PSRA.DW.W	PSRA.DB.B0	
101				
110				
111				



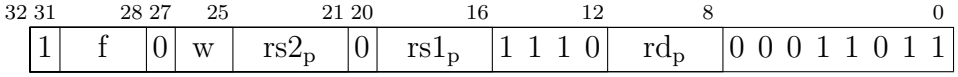
RV32 f	w 00	01	10	11
0000	PADD.DH	PADD.DW	PADD.DB	ADDD
0001				
0010	PSADD.DH	PSADD.DW	PSADD.DB	
0011	PAADD.DH	PAADD.DW	PAADD.DB	
0100				
0101				
0110	PSADDU.DH	PSADDU.DW	PSADDU.DB	
0111	PAADDU.DH	PAADDU.DW	PAADDU.DB	
1000	PSUB.DH	PSUB.DW	PSUB.DB	SUBD
1001	PDIF.DH	{ PDIF.DW }	PDIF.DB	
1010	PSSUB.DH	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	



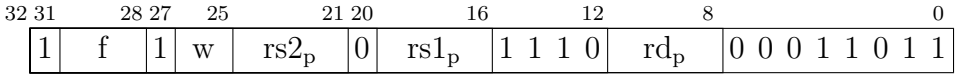
RV32 f	w 00	01	10	11
000				
001				
010	PSH1ADD.DH	PSH1ADD.DW		
011	PSSH1SADD.DH	PSSH1SADD.DW		
100				
101				
110				
111				



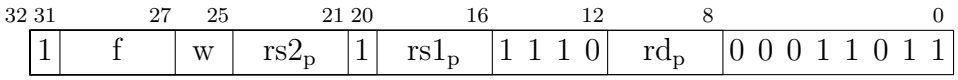
RV32 f	w 00	01	10	11
000	{ PSLL.DH }	{ PSLL.DW }	{ PSLL.DB }	
001				
010	{ PSSHL.DH }	{ PSSHL.DW }	{ PSSHL.DB }	
011	{ PSSHLR.DH }	{ PSSHLR.DW }	{ PSSHLR.DB }	
100				
101				
110	{ PSSHA.DH }	{ PSSHA.DW }	{ PSSHA.DB }	
111	{ PSSHAR.DH }	{ PSSHAR.DW }	{ PSSHAR.DB }	



RV32	w			
f	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				



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



RV32	w			
f	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	