D = immediate (I=1). D = register ** If #S and cogex, PC += signed(S). If #S and hubex, PC += signed(S*4). If S, PC = register S. * +1 if crosses hub long +1 if crosses hub lof * +1 if crosses hub long +1 if crosses hub lof Data not forwarded.

0	- Alias -	- Group -	- Encoding -	- Assembly Syntax -		- Description -	- Cogex Cycles -	- Hubex Cycles -	- Cogex Cycles -	- Hubex Cycles -	- Register Write -	- Hub R/W -	- Stack R/W -
1	.	Miscellaneous	0000 0000000 000 000000000 000000000	NOP		No operation.	2	same	2	same			
2	.	Math and Logic	EEEE 0000000 CZI DDDDDDDDD SSSSSSSSS	ROR	D,{#}S {WC/WZ/WCZ}	Rotate right. D = [31:0] of ({D[31:0], D[31:0]} >> S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[0]. *	2	same	2	same	D		
3	.	Math and Logic	EEEE 0000001 CZI DDDDDDDDD SSSSSSSSS	ROL	D,{#}S {WC/WZ/WCZ}	Rotate left. D = [63:32] of ({D[31:0], D[31:0]} << S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[31]. *	2	same	2	same	D		
4	.	Math and Logic	EEEE 0000010 CZI DDDDDDDDD SSSSSSSSS	SHR	D,{#}S {WC/WZ/WCZ}	Shift right. D = [31:0] of ({32'b0, D[31:0]} >> S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[0]. *	2	same	2	same	D		
5	.	Math and Logic	EEEE 0000011 CZI DDDDDDDDD SSSSSSSSS	SHL	D,{#}S {WC/WZ/WCZ}	Shift left. D = [63:32] of ({D[31:0], 32'b0} << S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[31]. *	2	same	2	same	D		
6	.	Math and Logic	EEEE 0000100 CZI DDDDDDDDD SSSSSSSSS	RCR	D,{#}S {WC/WZ/WCZ}	Rotate carry right. D = [31:0] of ({{32{C}}, D[31:0]} >> S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[0]. *	2	same	2	same	D		
7	.	Math and Logic	EEEE 0000101 CZI DDDDDDDDD SSSSSSSSS	RCL	D,{#}S {WC/WZ/WCZ}	Rotate carry left. D = [63:32] of ({D[31:0], {32{C}}} << S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[31]. *	2	same	2	same	D		
8	.	Math and Logic	EEEE 0000110 CZI DDDDDDDDD SSSSSSSSS	SAR	D,{#}S {WC/WZ/WCZ}	Shift arithmetic right. D = [31:0] of ({{32{D[31]}}, D[31:0]} >> S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[0]. *	2	same	2	same	D		
9	.	Math and Logic	EEEE 0000111 CZI DDDDDDDDD SSSSSSSSS	SAL	D,{#}S {WC/WZ/WCZ}	Shift arithmetic left. D = [63:32] of ({D[31:0], {32{D[0]}}} << S[4:0]). C = last bit shifted out if S[4:0] > 0, else D[31]. *	2	same	2	same	D		
10	.	Math and Logic	EEEE 0001000 CZI DDDDDDDDD SSSSSSSSS	ADD	D,{#}S {WC/WZ/WCZ}	Add S into D. D = D + S. C = carry of (D + S). *	2	same	2	same	D		
11	.	Math and Logic	EEEE 0001001 CZI DDDDDDDDD SSSSSSSSS	ADDX	D,{#}S {WC/WZ/WCZ}	Add (S + C) into D, extended. D = D + S + C. C = carry of (D + S + C). Z = Z AND (result == 0).	2	same	2	same	D		
12	.	Math and Logic	EEEE 0001010 CZI DDDDDDDDD SSSSSSSSS	ADDS	D,{#}S {WC/WZ/WCZ}	Add S into D, signed. D = D + S. C = correct sign of (D + S). *	2	same	2	same	D		
13	.	Math and Logic	EEEE 0001011 CZI DDDDDDDDD SSSSSSSSS	ADDSX	D,{#}S {WC/WZ/WCZ}	Add (S + C) into D, signed and extended. D = D + S + C. C = correct sign of (D + S + C). Z = Z AND (result == 0).	2	same	2	same	D		
14	.	Math and Logic	EEEE 0001100 CZI DDDDDDDDD SSSSSSSSS	SUB	D,{#}S {WC/WZ/WCZ}	Subtract S from D. D = D - S. C = borrow of (D - S). *	2	same	2	same	D		
15	.	Math and Logic	EEEE 0001101 CZI DDDDDDDDD SSSSSSSSS	SUBX	D,{#}S {WC/WZ/WCZ}	Subtract (S + C) from D, extended. D = D - (S + C). C = borrow of (D - (S + C)). Z = Z AND (result == 0).	2	same	2	same	D		
16	.	Math and Logic	EEEE 0001110 CZI DDDDDDDDD SSSSSSSSS	SUBS	D,{#}S {WC/WZ/WCZ}	Subtract S from D, signed. D = D - S. C = correct sign of (D - S). *	2	same	2	same	D		
17	.	Math and Logic	EEEE 0001111 CZI DDDDDDDDD SSSSSSSSS	SUBSX	D,{#}S {WC/WZ/WCZ}	Subtract (S + C) from D, signed and extended. D = D - (S + C). C = correct sign of (D - (S + C)). Z = Z AND (result == 0).	2	same	2	same	D		
18	.	Math and Logic	EEEE 0010000 CZI DDDDDDDDD SSSSSSSSS	CMP	D,{#}S {WC/WZ/WCZ}	Compare D to S. C = borrow of (D - S). Z = (D == S).	2	same	2	same			
19	.	Math and Logic	EEEE 0010001 CZI DDDDDDDDD SSSSSSSSS	CMPX	D,{#}S {WC/WZ/WCZ}	Compare D to (S + C), extended. C = borrow of (D - (S + C)). Z = Z AND (D == S + C).	2	same	2	same			
20	.	Math and Logic	EEEE 0010010 CZI DDDDDDDDD SSSSSSSSS	CMPS	D,{#}S {WC/WZ/WCZ}	Compare D to S, signed. C = correct sign of (D - S). Z = (D == S).	2	same	2	same			
21	.	Math and Logic	EEEE 0010011 CZI DDDDDDDDD SSSSSSSSS	CMPSX	D,{#}S {WC/WZ/WCZ}	Compare D to (S + C), signed and extended. C = correct sign of (D - (S + C)). Z = Z AND (D == S + C).	2	same	2	same			
22	.	Math and Logic	EEEE 0010100 CZI DDDDDDDDD SSSSSSSSS	CMPR	D,{#}S {WC/WZ/WCZ}	Compare S to D (reverse). C = borrow of (S - D). Z = (D == S).	2	same	2	same			
23	.	Math and Logic	EEEE 0010101 CZI DDDDDDDDD SSSSSSSSS	CMPM	D,{#}S {WC/WZ/WCZ}	Compare D to S, get MSB of difference into C. C = MSB of (D - S). Z = (D == S).	2	same	2	same			
24	.	Math and Logic	EEEE 0010110 CZI DDDDDDDDD SSSSSSSSS	SUBR	D,{#}S {WC/WZ/WCZ}	Subtract D from S (reverse). D = S - D. C = borrow of (S - D). *	2	same	2	same	D		
25	.	Math and Logic	EEEE 0010111 CZI DDDDDDDDD SSSSSSSSS	CMPSUB	D,{#}S {WC/WZ/WCZ}	Compare and subtract S from D if D >= S. If D => S then D = D - S and C = 1, else D same and C = 0. *	2	same	2	same	D		
26	.	Math and Logic	EEEE 0011000 CZI DDDDDDDDD SSSSSSSSS	FGE	D,{#}S {WC/WZ/WCZ}	Force D >= S. If D < S then D = S and C = 1, else D same and C = 0. *	2	same	2	same	D		
27	.	Math and Logic	EEEE 0011001 CZI DDDDDDDDD SSSSSSSSS	FLE	D,{#}S {WC/WZ/WCZ}	Force D <= S. If D > S then D = S and C = 1, else D same and C = 0. *	2	same	2	same	D		
28	.	Math and Logic	EEEE 0011010 CZI DDDDDDDDD SSSSSSSSS	FGES	D,{#}S {WC/WZ/WCZ}	Force D >= S, signed. If D < S then D = S and C = 1, else D same and C = 0. *	2	same	2	same	D		
29	.	Math and Logic	EEEE 0011011 CZI DDDDDDDDD SSSSSSSSS	FLES	D,{#}S {WC/WZ/WCZ}	Force D <= S, signed. If D > S then D = S and C = 1, else D same and C = 0. *	2	same	2	same	D		
30	.	Math and Logic	EEEE 0011100 CZI DDDDDDDDD SSSSSSSSS	SUMC	D,{#}S {WC/WZ/WCZ}	Sum +/-S into D by C. If C = 1 then D = D - S, else D = D + S. C = correct sign of (D +/- S). *	2	same	2	same	D		
31	.	Math and Logic	EEEE 0011101 CZI DDDDDDDDD SSSSSSSSS	SUMNC	D,{#}S {WC/WZ/WCZ}	Sum +/-S into D by !C. If C = 0 then D = D - S, else D = D + S. C = correct sign of (D +/- S). *	2	same	2	same	D		
32	.	Math and Logic	EEEE 0011110 CZI DDDDDDDDD SSSSSSSSS	SUMZ	D,{#}S {WC/WZ/WCZ}	Sum +/-S into D by Z. If Z = 1 then D = D - S, else D = D + S. C = correct sign of (D +/- S). *	2	same	2	same	D		
33	.	Math and Logic	EEEE 0011111 CZI DDDDDDDDD SSSSSSSSS	SUMNZ	D,{#}S {WC/WZ/WCZ}	Sum +/-S into D by !Z. If Z = 0 then D = D - S, else D = D + S. C = correct sign of (D +/- S). *	2	same	2	same	D		
34	.	Math and Logic	EEEE 0100000 CZI DDDDDDDDD SSSSSSSSS	TESTB	D,{#}S WC/WZ	Test bit S[4:0] of D, write to C/Z. C/Z = D[S[4:0]].	2	same	2	same			
35	.	Math and Logic	EEEE 0100001 CZI DDDDDDDDD SSSSSSSSS	TESTBN	D,{#}S WC/WZ	Test bit S[4:0] of !D, write to C/Z. C/Z = !D[S[4:0]].	2	same	2	same			
36	.	Math and Logic	EEEE 0100010 CZI DDDDDDDDD SSSSSSSSS	TESTB	D,{#}S ANDC/ANDZ	Test bit S[4:0] of D, AND into C/Z. C/Z = C/Z AND D[S[4:0]].	2	same	2	same			
37	.	Math and Logic	EEEE 0100011 CZI DDDDDDDDD SSSSSSSSS	TESTBN	D,{#}S ANDC/ANDZ	Test bit S[4:0] of !D, AND into C/Z. C/Z = C/Z AND !D[S[4:0]].	2	same	2	same			
38	.	Math and Logic	EEEE 0100100 CZI DDDDDDDDD SSSSSSSSS	TESTB	D,{#}S ORC/ORZ	Test bit S[4:0] of D, OR into C/Z. C/Z = C/Z OR D[S[4:0]].	2	same	2	same			
39	.	Math and Logic	EEEE 0100101 CZI DDDDDDDDD SSSSSSSSS	TESTBN	D,{#}S ORC/ORZ	Test bit S[4:0] of !D, OR into C/Z. C/Z = C/Z OR !D[S[4:0]].	2	same	2	same			
40	.	Math and Logic	EEEE 0100110 CZI DDDDDDDDD SSSSSSSSS	TESTB	D,{#}S XORC/XORZ	Test bit S[4:0] of D, XOR into C/Z. C/Z = C/Z XOR D[S[4:0]].	2	same	2	same			
41	.	Math and Logic	EEEE 0100111 CZI DDDDDDDDD SSSSSSSSS	TESTBN	D,{#}S XORC/XORZ	Test bit S[4:0] of !D, XOR into C/Z. C/Z = C/Z XOR !D[S[4:0]].	2	same	2	same			
42	.	Math and Logic	EEEE 0100000 CZI DDDDDDDDD SSSSSSSSS	BITL	D,{#}S {WCZ}	Bit S[4:0] of D = 0, C,Z = D[S[4:0]].	2	same	2	same	D		
43	.	Math and Logic	EEEE 0100001 CZI DDDDDDDDD SSSSSSSSS	BITH	D,{#}S {WCZ}	Bit S[4:0] of D = 1, C,Z = D[S[4:0]].	2	same	2	same	D		
44	.	Math and Logic	EEEE 0100010 CZI DDDDDDDDD SSSSSSSSS	BITC	D,{#}S {WCZ}	Bit S[4:0] of D = C, C,Z = D[S[4:0]].	2	same	2	same	D		
45	.	Math and Logic	EEEE 0100011 CZI DDDDDDDDD SSSSSSSSS	BITNC	D,{#}S {WCZ}	Bit S[4:0] of D = !C, C,Z = D[S[4:0]].	2	same	2	same	D		
46	.	Math and Logic	EEEE 0100100 CZI DDDDDDDDD SSSSSSSSS	BITZ	D,{#}S {WCZ}	Bit S[4:0] of D = Z, C,Z = D[S[4:0]].	2	same	2	same	D		
47	.	Math and Logic	EEEE 0100101 CZI DDDDDDDDD SSSSSSSSS	BITNZ	D,{#}S {WCZ}	Bit S[4:0] of D = !Z, C,Z = D[S[4:0]].	2	same	2	same	D		
48	.	Math and Logic	EEEE 0100110 CZI DDDDDDDDD SSSSSSSSS	BITRND	D,{#}S {WCZ}	Bit S[4:0] of D = RND, C,Z = D[S[4:0]].	2	same	2	same	D		
49	.	Math and Logic	EEEE 0100111 CZI DDDDDDDDD SSSSSSSSS	BITNOT	D,{#}S {WCZ}	Bit S[4:0] of D = !bit, C,Z = D[S[4:0]].	2	same	2	same	D		
50	.	Math and Logic	EEEE 0101000 CZI DDDDDDDDD SSSSSSSSS	AND	D,{#}S {WC/WZ/WCZ}	AND S into D. D = D & S. C = parity of result. *	2	same	2	same	D		
51	.	Math and Logic	EEEE 0101001 CZI DDDDDDDDD SSSSSSSSS	ANDN	D,{#}S {WC/WZ/WCZ}	AND !S into D. D = D & !S. C = parity of result. *	2	same	2	same	D		
52	.	Math and Logic	EEEE 0101010 CZI DDDDDDDDD SSSSSSSSS	OR	D,{#}S {WC/WZ/WCZ}	OR S into D. D = D | S. C = parity of result. *	2	same	2	same	D		
53	.	Math and Logic	EEEE 0101011 CZI DDDDDDDDD SSSSSSSSS	XOR	D,{#}S {WC/WZ/WCZ}	XOR S into D. D = D ^ S. C = parity of result. *	2	same	2	same	D		
54	.	Math and Logic	EEEE 0101100 CZI DDDDDDDDD SSSSSSSSS	MUXC	D,{#}S {WC/WZ/WCZ}	Mux C into each D bit that is '1' in S. D = (!S & D) | (S & {32{ C}}). C = parity of result. *	2	same	2	same	D		
55	.	Math and Logic	EEEE 0101101 CZI DDDDDDDDD SSSSSSSSS	MUXNC	D,{#}S {WC/WZ/WCZ}	Mux !C into each D bit that is '1' in S. D = (!S & D) | (S & {32{!C}}). C = parity of result. *	2	same	2	same	D		
56	.	Math and Logic	EEEE 0101110 CZI DDDDDDDDD SSSSSSSSS	MUXZ	D,{#}S {WC/WZ/WCZ}	Mux Z into each D bit that is '1' in S. D = (!S & D) | (S & {32{ Z}}). C = parity of result. *	2	same	2	same	D		
57	.	Math and Logic	EEEE 0101111 CZI DDDDDDDDD SSSSSSSSS	MUXNZ	D,{#}S {WC/WZ/WCZ}	Mux !Z into each D bit that is '1' in S. D = (!S & D) | (S & {32{!Z}}). C = parity of result. *	2	same	2	same	D		
58	.	Math and Logic	EEEE 0110000 CZI DDDDDDDDD SSSSSSSSS	MOV	D,{#}S {WC/WZ/WCZ}	Move S into D. D = S. C = S[31]. *	2	same	2	same	D		
59	.	Math and Logic	EEEE 0110001 CZI DDDDDDDDD SSSSSSSSS	NOT	D,{#}S {WC/WZ/WCZ}	Get !S into D. D = !S. C = !S[31]. *	2	same	2	same	D		
60	alias	Math and Logic	EEEE 0110001 CZ0 DDDDDDDDD DDDDDDDDD	NOT	D {WC/WZ/WCZ}	Get !D into D. D = !D. C = !D[31]. *	2	same	2	same	D		
61	.	Math and Logic	EEEE 0110010 CZI DDDDDDDDD SSSSSSSSS	ABS	D,{#}S {WC/WZ/WCZ}	Get absolute value of S into D. D = ABS(S). C = S[31]. *	2	same	2	same	D		
62	alias	Math and Logic	EEEE 0110010 CZ0 DDDDDDDDD DDDDDDDDD	ABS	D {WC/WZ/WCZ}	Get absolute value of D into D. D = ABS(D). C = D[31]. *	2	same	2	same	D		
63	.	Math and Logic	EEEE 0110011 CZI DDDDDDDDD SSSSSSSSS	NEG	D,{#}S {WC/WZ/WCZ}	Negate S into D. D = -S. C = MSB of result. *	2	same	2	same	D		
64	alias	Math and Logic	EEEE 0110011 CZ0 DDDDDDDDD DDDDDDDDD	NEG	D {WC/WZ/WCZ}	Negate D. D = -D. C = MSB of result. *	2	same	2	same	D		
65	.	Math and Logic	EEEE 0110100 CZI DDDDDDDDD SSSSSSSSS	NEGC	D,{#}S {WC/WZ/WCZ}	Negate S by C into D. If C = 1 then D = -S, else D = S. C = MSB of result. *	2	same	2	same	D		
66	alias	Math and Logic	EEEE 0110100 CZ0 DDDDDDDDD DDDDDDDDD	NEGC	D {WC/WZ/WCZ}	Negate D by C. If C = 1 then D = -D, else D = D. C = MSB of result. *	2	same	2	same	D		
67	.	Math and Logic	EEEE 0110101 CZI DDDDDDDDD SSSSSSSSS	NEGNC	D,{#}S {WC/WZ/WCZ}	Negate S by !C into D. If C = 0 then D = -S, else D = S. C = MSB of result. *	2	same	2	same	D		
68	alias	Math and Logic	EEEE 0110101 CZ0 DDDDDDDDD DDDDDDDDD	NEGNC	D {WC/WZ/WCZ}	Negate D by !C. If C = 0 then D = -D, else D = D. C = MSB of result. *	2	same	2	same	D		
69	.	Math and Logic	EEEE 0110110 CZI DDDDDDDDD SSSSSSSSS	NEGZ	D,{#}S {WC/WZ/WCZ}	Negate S by Z into D. If Z = 1 then D = -S, else D = S. C = MSB of result. *	2	same	2	same	D		
70	alias	Math and Logic	EEEE 0110110 CZ0 DDDDDDDDD DDDDDDDDD	NEGZ	D {WC/WZ/WCZ}	Negate D by Z. If Z = 1 then D = -D, else D = D. C = MSB of result. *	2	same	2	same	D		
71	.	Math and Logic	EEEE 0110111 CZI DDDDDDDDD SSSSSSSSS	NEGNZ	D,{#}S {WC/WZ/WCZ}	Negate S by !Z into D. If Z = 0 then D = -S, else D = S. C = MSB of result. *	2	same	2	same	D		
72	alias	Math and Logic	EEEE 0110111 CZ0 DDDDDDDDD DDDDDDDDD	NEGNZ	D {WC/WZ/WCZ}	Negate D by !Z. If Z = 0 then D = -D, else D = D. C = MSB of result. *	2	same	2	same	D		
73	.	Math and Logic	EEEE 0111000 CZI DDDDDDDDD SSSSSSSSS	INCMOD	D,{#}S {WC/WZ/WCZ}	Increment with modulus. If D = S then D = 0 and C = 1, else D = D + 1 and C = 0. *	2	same	2	same	D		
74	.	Math and Logic	EEEE 0111001 CZI DDDDDDDDD SSSSSSSSS	DECMOD	D,{#}S {WC/WZ/WCZ}	Decrement with modulus. If D = 0 then D = S and C = 1, else D = D - 1 and C = 0. *	2	same	2	same	D		
75	.	Math and Logic	EEEE 0111010 CZI DDDDDDDDD SSSSSSSSS	ZEROX	D,{#}S {WC/WZ/WCZ}	Zero-extend D above bit S[4:0]. C = MSB of result. *	2	same	2	same	D		
76	.	Math and Logic	EEEE 0111011 CZI DDDDDDDDD SSSSSSSSS	SIGNX	D,{#}S {WC/WZ/WCZ}	Sign-extend D from bit S[4:0]. C = MSB of result. *	2	same	2	same	D		
77	.	Math and Logic	EEEE 0111100 CZI DDDDDDDDD SSSSSSSSS	ENCOD	D,{#}S {WC/WZ/WCZ}	Get bit position of top-most '1' in S into D. D = position of top '1' in S (0..31). C = (S != 0). *	2	same	2	same	D		
78	alias	Math and Logic	EEEE 0111100 CZ0 DDDDDDDDD DDDDDDDDD	ENCOD	D {WC/WZ/WCZ}	Get bit position of top-most '1' in D into D. D = position of top '1' in S (0..31). C = (S != 0). *	2	same	2	same	D		
79	.	Math and Logic	EEEE 0111101 CZI DDDDDDDDD SSSSSSSSS	ONES	D,{#}S {WC/WZ/WCZ}	Get number of '1's in S into D. D = number of '1's in S (0..32). C = LSB of result. *	2	same	2	same	D		
80	alias	Math and Logic	EEEE 0111101 CZ0 DDDDDDDDD DDDDDDDDD	ONES	D {WC/WZ/WCZ}	Get number of '1's in D into D. D = number of '1's in S (0..32). C = LSB of result. *	2	same	2	same	D		
81	.	Math and Logic	EEEE 0111110 CZI DDDDDDDDD SSSSSSSSS	TEST	D,{#}S {WC/WZ/WCZ}	Test D with S. C = parity of (D & S). Z = ((D & S) == 0).	2	same	2	same			
82	alias	Math and Logic	EEEE 0111110 CZ0 DDDDDDDDD DDDDDDDDD	TEST	D {WC/WZ/WCZ}	Test D. C = parity of D. Z = (D == 0).	2	same	2	same			
83	.	Math and Logic	EEEE 0111111 CZI DDDDDDDDD SSSSSSSSS	TESTN	D,{#}S {WC/WZ/WCZ}	Test D with !S. C = parity of (D & !S). Z = ((D & !S) == 0).	2	same	2	same			
84	.	Math and Logic	EEEE 100000N NNI DDDDDDDDD SSSSSSSSS	SETNIB	D,{#}S,#N	Set S[3:0] into nibble N in D, keeping rest of D same.	2	same	2	same	D		
85	alias	Math and Logic	EEEE 1000000 00I 000000000 SSSSSSSSS	SETNIB	{#}S	Set S[3:0] into nibble established by prior ALTSN instruction.	2	same	2	same	D		
86	.	Math and Logic	EEEE 100001N NNI DDDDDDDDD SSSSSSSSS	GETNIB	D,{#}S,#N	Get nibble N of S into D. D = {28'b0, S.NIBBLE[N]}.	2	same	2	same	D		
87	alias	Math and Logic	EEEE 1000010 000 DDDDDDDDD 000000000	GETNIB	D	Get nibble established by prior ALTGN instruction into D.	2	same	2	same	D		
88	.	Math and Logic	EEEE 100010N NNI DDDDDDDDD SSSSSSSSS	ROLNIB	D,{#}S,#N	Rotate-left nibble N of S into D. D = {D[27:0], S.NIBBLE[N]}.	2	same	2	same	D		
89	alias	Math and Logic	EEEE 1000100 000 DDDDDDDDD 000000000	ROLNIB	D	Rotate-left nibble established by prior ALTGN instruction into D.	2	same	2	same	D		
90	.	Math and Logic	EEEE 1000110 NNI DDDDDDDDD SSSSSSSSS	SETBYTE	D,{#}S,#N	Set S[7:0] into byte N in D, keeping rest of D same.	2	same	2	same	D		
91	alias	Math and Logic	EEEE 1000110 00I 000000000 SSSSSSSSS	SETBYTE	{#}S	Set S[7:0] into byte established by prior ALTSB instruction.	2	same	2	same	D		
92	.	Math and Logic	EEEE 1000111 NNI DDDDDDDDD SSSSSSSSS	GETBYTE	D,{#}S,#N	Get byte N of S into D. D = {24'b0, S.BYTE[N]}.	2	same	2	same	D		
93	alias	Math and Logic	EEEE 1000111 000 DDDDDDDDD 000000000	GETBYTE	D	Get byte established by prior ALTGB instruction into D.	2	same	2	same	D		
94	.	Math and Logic	EEEE 1001000 NNI DDDDDDDDD SSSSSSSSS	ROLBYTE	D,{#}S,#N	Rotate-left byte N of S into D. D = {D[23:0], S.BYTE[N]}.	2	same	2	same	D		
95	alias	Math and Logic	EEEE 1001000 000 DDDDDDDDD 000000000	ROLBYTE	D	Rotate-left byte established by prior ALTGB instruction into D.	2	same	2	same	D		
96	.	Math and Logic	EEEE 1001001 0NI DDDDDDDDD SSSSSSSSS	SETWORD	D,{#}S,#N	Set S[15:0] into word N in D, keeping rest of D same.	2	same	2	same	D		
97	alias	Math and Logic	EEEE 1001001 00I 000000000 SSSSSSSSS	SETWORD	{#}S	Set S[15:0] into word established by prior ALTSW instruction.	2	same	2	same	D		
98	.	Math and Logic	EEEE 1001001 1NI DDDDDDDDD SSSSSSSSS	GETWORD	D,{#}S,#N	Get word N of S into D. D = {16'b0, S.WORD[N]}.	2	same	2	same	D		
99	alias	Math and Logic	EEEE 1001001 100 DDDDDDDDD 000000000	GETWORD	D	Get word established by prior ALTGW instruction into D.	2	same	2	same	D		
100	.	Math and Logic	EEEE 1001010 0NI DDDDDDDDD SSSSSSSSS	ROLWORD	D,{#}S,#N	Rotate-left word N of S into D. D = {D[15:0], S.WORD[N]}.	2	same	2	same	D		
101	alias	Math and Logic	EEEE 1001010 000 DDDDDDDDD 000000000	ROLWORD	D	Rotate-left word established by prior ALTGW instruction into D.	2	same	2	same	D		
102	.	Register Indirection	EEEE 1001010 10I DDDDDDDDD SSSSSSSSS	ALTSN	D,{#}S	Alter subsequent SETNIB instruction. Next D field = (D[11:3] + S) & $1FF, N field = D[2:0]. D += sign-extended S[17:9].	2	same	2	same	D		
103	alias	Register Indirection	EEEE 1001010 101 DDDDDDDDD 000000000	ALTSN	D	Alter subsequent SETNIB instruction. Next D field = D[11:3], N field = D[2:0].	2	same	2	same	D		
104	.	Register Indirection	EEEE 1001010 11I DDDDDDDDD SSSSSSSSS	ALTGN	D,{#}S	Alter subsequent GETNIB/ROLNIB instruction. Next S field = (D[11:3] + S) & $1FF, N field = D[2:0]. D += sign-extended S[17:9].	2	same	2	same	D		

105	alias	Register Indirection	EEEE 1001010 111 DDDDDDDDD 000000000	ALTGN	D			Alter subsequent GETNIB/ROLNIB instruction. Next S field = D[11:3], N field = D[2:0].	2	same	2	same	D		
106	.	Register Indirection	EEEE 1001011 00I DDDDDDDDD SSSSSSSSS	ALTSB	D,{#}S			Alter subsequent SETBYTE instruction. Next D field = (D[10:2] + S) & $1FF, N field = D[1:0]. D += sign-extended S[17:9].	2	same	2	same	D		
107	alias	Register Indirection	EEEE 1001011 00I DDDDDDDDD 000000000	ALTSB	D			Alter subsequent SETBYTE instruction. Next D field = D[10:2], N field = D[1:0].	2	same	2	same	D		
108	.	Register Indirection	EEEE 1001011 01I DDDDDDDDD SSSSSSSSS	ALTGB	D,{#}S			Alter subsequent GETBYTE/ROLBYTE instruction. Next S field = (D[10:2] + S) & $1FF, N field = D[1:0]. D += sign-extended S[17:9].	2	same	2	same	D		
109	alias	Register Indirection	EEEE 1001011 01I DDDDDDDDD 000000000	ALTGB	D			Alter subsequent GETBYTE/ROLBYTE instruction. Next S field = D[10:2], N field = D[1:0].	2	same	2	same	D		
110	.	Register Indirection	EEEE 1001011 10I DDDDDDDDD SSSSSSSSS	ALTSW	D,{#}S			Alter subsequent SETWORD instruction. Next D field = (D[9:1] + S) & $1FF, N field = D[0]. D += sign-extended S[17:9].	2	same	2	same	D		
111	alias	Register Indirection	EEEE 1001011 10I DDDDDDDDD 000000000	ALTSW	D			Alter subsequent SETWORD instruction. Next D field = D[9:1], N field = D[0].	2	same	2	same	D		
112	.	Register Indirection	EEEE 1001011 11I DDDDDDDDD SSSSSSSSS	ALTGW	D,{#}S			Alter subsequent GETWORD/ROLWORD instruction. Next S field = ((D[9:1] + S) & $1FF), N field = D[0]. D += sign-extended S[17:9].	2	same	2	same	D		
113	alias	Register Indirection	EEEE 1001011 111 DDDDDDDDD 000000000	ALTGW	D			Alter subsequent GETWORD/ROLWORD instruction. Next S field = D[9:1], N field = D[0].	2	same	2	same	D		
114	.	Register Indirection	EEEE 1001100 00I DDDDDDDDD SSSSSSSSS	ALTR	D,{#}S			Alter result register address (normally D field) of next instruction to (D + S) & $1FF. D += sign-extended S[17:9].	2	same	2	same	D		
115	alias	Register Indirection	EEEE 1001100 00I DDDDDDDDD 000000000	ALTR	D			Alter result register address (normally D field) of next instruction to D[8:0].	2	same	2	same	D		
116	.	Register Indirection	EEEE 1001100 01I DDDDDDDDD SSSSSSSSS	ALTD	D,{#}S			Alter D field of next instruction to (D + S) & $1FF. D += sign-extended S[17:9].	2	same	2	same	D		
117	alias	Register Indirection	EEEE 1001100 01I DDDDDDDDD 000000000	ALTD	D			Alter D field of next instruction to D[8:0].	2	same	2	same	D		
118	.	Register Indirection	EEEE 1001100 10I DDDDDDDDD SSSSSSSSS	ALTS	D,{#}S			Alter S field of next instruction to (D + S) & $1FF. D += sign-extended S[17:9].	2	same	2	same	D		
119	alias	Register Indirection	EEEE 1001100 10I DDDDDDDDD 000000000	ALTS	D			Alter S field of next instruction to D[8:0].	2	same	2	same	D		
120	.	Register Indirection	EEEE 1001100 11I DDDDDDDDD SSSSSSSSS	ALTB	D,{#}S			Alter D field of next instruction to (D[13:5] + S) & $1FF. D += sign-extended S[17:9].	2	same	2	same	D		
121	alias	Register Indirection	EEEE 1001100 111 DDDDDDDDD 000000000	ALTB	D			Alter D field of next instruction to D[13:5].	2	same	2	same	D		
122	.	Register Indirection	EEEE 1001101 00I DDDDDDDDD SSSSSSSSS	ALTI	D,{#}S			Substitute next instruction's I/R/D/S fields with fields from D, per S. Modify D per S.	2	same	2	same	D		
123	alias	Register Indirection	EEEE 1001101 00I DDDDDDDDD 101100100	ALTI	D			Execute D in place of next instruction. D stays same.	2	same	2	same	D		
124	.	Math and Logic	EEEE 1001101 01I DDDDDDDDD SSSSSSSSS	SETR	D,{#}S			Set R field of D to S[8:0]. D = {D[31:28], S[8:0], D[18:0]}.	2	same	2	same	D		
125	.	Math and Logic	EEEE 1001101 10I DDDDDDDDD SSSSSSSSS	SETD	D,{#}S			Set D field of D to S[8:0]. D = {D[31:18], S[8:0], D[8:0]}.	2	same	2	same	D		
126	.	Math and Logic	EEEE 1001101 11I DDDDDDDDD SSSSSSSSS	SETS	D,{#}S			Set S field of D to S[8:0]. D = {D[31:9], S[8:0]}.	2	same	2	same	D		
127	.	Math and Logic	EEEE 1001110 00I DDDDDDDDD SSSSSSSSS	DECOD	D,{#}S			Decode S[4:0] into D. D = 1 << S[4:0].	2	same	2	same	D		
128	alias	Math and Logic	EEEE 1001110 000 DDDDDDDDD DDDDDDDDD	DECOD	D			Decode D[4:0] into D. D = 1 << D[4:0].	2	same	2	same	D		
129	.	Math and Logic	EEEE 1001110 01I DDDDDDDDD SSSSSSSSS	BMASK	D,{#}S			Get LSB-justified bit mask of size (S[4:0] + 1) into D. D = ($0000_0002 << S[4:0]) - 1.	2	same	2	same	D		
130	alias	Math and Logic	EEEE 1001110 010 DDDDDDDDD DDDDDDDDD	BMASK	D			Get LSB-justified bit mask of size (D[4:0] + 1) into D. D = ($0000_0002 << D[4:0]) - 1.	2	same	2	same	D		
131	.	Math and Logic	EEEE 1001110 10I DDDDDDDDD SSSSSSSSS	CRCBIT	D,{#}S			Iterate CRC value in D using C and polynomial in S. If (C XOR D[0]) then D = (D >> 1) XOR S, else D = (D >> 1).	2	same	2	same	D		
132	.	Math and Logic	EEEE 1001110 11I DDDDDDDDD SSSSSSSSS	CRCNIB	D,{#}S			Iterate CRC value in D using Q[31:28] and polynomial in S. Like CRCBIT, but 4x. Q = Q << 4. Use SETQ+CRCNIB+CRCNIB+CRCNIB...	2	same	2	same	D		
133	.	Math and Logic	EEEE 1001111 00I DDDDDDDDD SSSSSSSSS	MUXNITS	D,{#}S			For each non-zero bit pair in S, copy that bit pair into the corresponding D bits, else leave that D bit pair the same.	2	same	2	same	D		
134	.	Math and Logic	EEEE 1001111 01I DDDDDDDDD SSSSSSSSS	MUXNIBS	D,{#}S			For each non-zero nibble in S, copy that nibble into the corresponding D nibble, else leave that D nibble the same.	2	same	2	same	D		
135	.	Math and Logic	EEEE 1001111 10I DDDDDDDDD SSSSSSSSS	MUXQ	D,{#}S			Used after SETQ. For each '1' bit in Q, copy the corresponding bit in S into D. D = (D & !Q) | (S & Q).	2	same	2	same	D		
136	.	Math and Logic	EEEE 1001111 111 DDDDDDDDD SSSSSSSSS	MOVBYTS	D,{#}S			Move bytes within D, per S. D = {D.BYTE[S[7:6]], D.BYTE[S[5:4]], D.BYTE[S[3:2]], D.BYTE[S[1:0]]}.	2	same	2	same	D		
137	.	Math and Logic	EEEE 1010000 0ZI DDDDDDDDD SSSSSSSSS	MUL	D,{#}S	{WZ}		D = unsigned (D[15:0] * S[15:0]). Z = (S == 0) | (D == 0).	2	same	2	same	D		
138	.	Math and Logic	EEEE 1010000 1ZI DDDDDDDDD SSSSSSSSS	MULS	D,{#}S	{WZ}		D = signed (D[15:0] * S[15:0]). Z = (S == 0) | (D == 0).	2	same	2	same	D		
139	.	Math and Logic	EEEE 1010001 0ZI DDDDDDDDD SSSSSSSSS	SCA	D,{#}S	{WZ}		Next instruction's S value = unsigned (D[15:0] * S[15:0]) >> 16. *	2	same	2	same			
140	.	Math and Logic	EEEE 1010001 1ZI DDDDDDDDD SSSSSSSSS	SCAS	D,{#}S	{WZ}		Next instruction's S value = signed (D[15:0] * S[15:0]) >> 14. In this scheme, $4000 = 1.0 and $C000 = -1.0. *	2	same	2	same			
141	.	Pixel Mixer	EEEE 1010010 00I DDDDDDDDD SSSSSSSSS	ADDPIX	D,{#}S			Add bytes of S into bytes of D, with $FF saturation.	7	same	7	same	D		
142	.	Pixel Mixer	EEEE 1010010 01I DDDDDDDDD SSSSSSSSS	MULPIX	D,{#}S			Multiply bytes of S into bytes of D, where $FF = 1.0.	7	same	7	same	D		
143	.	Pixel Mixer	EEEE 1010010 10I DDDDDDDDD SSSSSSSSS	BLNPIX	D,{#}S			Alpha-blend bytes of S into bytes of D, using SETPIV value.	7	same	7	same	D		
144	.	Pixel Mixer	EEEE 1010010 11I DDDDDDDDD SSSSSSSSS	MIXPIX	D,{#}S			Mix bytes of S into bytes of D, using SETPIX and SETPIV values.	7	same	7	same	D		
145	.	Events - Configuratio	EEEE 1010011 00I DDDDDDDDD SSSSSSSSS	ADDCT1	D,{#}S			Set CT1 event to trigger on CT = D + S. Adds S into D.	2	same	2	same	D		
146	.	Events - Configuratio	EEEE 1010011 01I DDDDDDDDD SSSSSSSSS	ADDCT2	D,{#}S			Set CT2 event to trigger on CT = D + S. Adds S into D.	2	same	2	same	D		
147	.	Events - Configuratio	EEEE 1010011 10I DDDDDDDDD SSSSSSSSS	ADDCT3	D,{#}S			Set CT3 event to trigger on CT = D + S. Adds S into D.	2	same	2	same	D		
148	.	Hub RAM - Write	EEEE 1010011 11I DDDDDDDDD SSSSSSSSS	WMLONG	D,{#}S/P			Write only non-$00 bytes in D[31:0] to hub address {#}S/PTRx. Prior SETQ/SETQ2 invokes cog/LUT block transfer.	3...10 *	3...20 *	3...18 *	3...38 *		Write	
149	.	Smart Pins	EEEE 1010100 C0I DDDDDDDDD SSSSSSSSS	RQPIN	D,{#}S		{WC}	Read smart pin S[5:0] result "Z" into D, don't acknowledge smart pin ("Q" in RQPIN means "quiet"). C = modal result.	2	same	2	same	D		
150	.	Smart Pins	EEEE 1010100 C1I DDDDDDDDD SSSSSSSSS	RDPIN	D,{#}S		{WC}	Read smart pin S[5:0] result "Z" into D, acknowledge smart pin. C = modal result.	2	same	2	same	D		
151	.	Lookup Table	EEEE 1010101 CZI DDDDDDDDD SSSSSSSSS	RDLUT	D,{#}S	{WC/WZ/WCZ}		Read LUT data from address S[8:0] into D. C = MSB of data. *	3	same	3	same	D		
152	.	Hub RAM - Read	EEEE 1010110 CZI DDDDDDDDD SSSSSSSSS	RDBYTE	D,{#}S/P	{WC/WZ/WCZ}		Read zero-extended byte from hub address {#}S/PTRx into D. C = MSB of byte. *	9...16	9...26	9...24	9...44	D	Read	
153	.	Hub RAM - Read	EEEE 1010111 CZI DDDDDDDDD SSSSSSSSS	RDWORD	D,{#}S/P	{WC/WZ/WCZ}		Read zero-extended word from hub address {#}S/PTRx into D. C = MSB of word. *	9...16 *	9...26 *	9...24 *	9...44 *	D	Read	
154	.	Hub RAM - Read	EEEE 1011000 CZI DDDDDDDDD SSSSSSSSS	RDLONG	D,{#}S/P	{WC/WZ/WCZ}		Read long from hub address {#}S/PTRx into D. C = MSB of long. * Prior SETQ/SETQ2 invokes cog/LUT block transfer.	9...16 *	9...26 *	9...24 *	9...44 *	D	Read	
155	alias	Hub RAM - Read	EEEE 1011000 CZI DDDDDDDDD 101011111	POPA	D	{WC/WZ/WCZ}		Read long from hub address --PTRA into D. C = MSB of long. *	9...16 *	9...26 *	9...24 *	9...44 *	D	Read	
156	alias	Hub RAM - Read	EEEE 1011000 CZI DDDDDDDDD 111011111	POPB	D	{WC/WZ/WCZ}		Read long from hub address --PTRB into D. C = MSB of long. *	9...16 *	9...26 *	9...24 *	9...44 *	D	Read	
157	.	Branch S - Call	EEEE 1011001 CZI DDDDDDDDD SSSSSSSSS	CALLD	D,{#}S	{WC/WZ/WCZ}		Call to S** by writing {C, Z, 10'b0, PC[19:0]} to D. C = S[31], Z = S[30].	4	13...20	4	13...28	D		
158	alias	Branch S - Resume	EEEE 1011001 110 111110000 111110001	RESI3				Resume from INT3. (CALLD $1F0,$1F1 WC,WZ)	4	13...20	4	13...28	D		
159	alias	Branch S - Resume	EEEE 1011001 110 111110010 111110011	RESI2				Resume from INT2. (CALLD $1F2,$1F3 WC,WZ)	4	13...20	4	13...28	D		
160	alias	Branch S - Resume	EEEE 1011001 110 111110100 111110101	RESI1				Resume from INT1. (CALLD $1F4,$1F5 WC,WZ)	4	13...20	4	13...28	D		
161	alias	Branch S - Resume	EEEE 1011001 110 111111110 111111111	RESI0				Resume from INT0. (CALLD $1FE,$1FF WC,WZ)	4	13...20	4	13...28	D		
162	alias	Branch S - Return	EEEE 1011001 110 111111111 111110001	RETI3				Return from INT3. (CALLD $1FF,$1F1 WC,WZ)	4	13...20	4	13...28	D		
163	alias	Branch S - Return	EEEE 1011001 110 111111111 111110011	RETI2				Return from INT2. (CALLD $1FF,$1F3 WC,WZ)	4	13...20	4	13...28	D		
164	alias	Branch S - Return	EEEE 1011001 110 111111111 111110101	RETI1				Return from INT1. (CALLD $1FF,$1F5 WC,WZ)	4	13...20	4	13...28	D		
165	alias	Branch S - Return	EEEE 1011001 110 111111111 111111111	RETI0				Return from INT0. (CALLD $1FF,$1FF WC,WZ)	4	13...20	4	13...28	D		
166	.	Branch S - Call	EEEE 1011010 0LI DDDDDDDDD SSSSSSSSS	CALLPA	{#}D,{#}S			Call to S** by pushing {C, Z, 10'b0, PC[19:0]} onto stack, copy D to PA.	4	13...20	4	13...28	PA		Push
167	.	Branch S - Call	EEEE 1011010 1LI DDDDDDDDD SSSSSSSSS	CALLPB	{#}D,{#}S			Call to S** by pushing {C, Z, 10'b0, PC[19:0]} onto stack, copy D to PB.	4	13...20	4	13...28	PB		Push
168	.	Branch S - Mod & Te	EEEE 1011011 00I DDDDDDDDD SSSSSSSSS	DJZ	D,{#}S			Decrement D and jump to S** if result is zero.	2 or 4	2 or 13...20	2 or 4	2 or 13...28	D		
169	.	Branch S - Mod & Te	EEEE 1011011 01I DDDDDDDDD SSSSSSSSS	DJNZ	D,{#}S			Decrement D and jump to S** if result is not zero.	2 or 4	2 or 13...20	2 or 4	2 or 13...28	D		
170	.	Branch S - Mod & Te	EEEE 1011011 10I DDDDDDDDD SSSSSSSSS	DJF	D,{#}S			Decrement D and jump to S** if result is $FFFF_FFFF.	2 or 4	2 or 13...20	2 or 4	2 or 13...28	D		
171	.	Branch S - Mod & Te	EEEE 1011011 11I DDDDDDDDD SSSSSSSSS	DJNF	D,{#}S			Decrement D and jump to S** if result is not $FFFF_FFFF.	2 or 4	2 or 13...20	2 or 4	2 or 13...28	D		
172	.	Branch S - Mod & Te	EEEE 1011100 00I DDDDDDDDD SSSSSSSSS	IJZ	D,{#}S			Increment D and jump to S** if result is zero.	2 or 4	2 or 13...20	2 or 4	2 or 13...28	D		
173	.	Branch S - Mod & Te	EEEE 1011100 01I DDDDDDDDD SSSSSSSSS	IJNZ	D,{#}S			Increment D and jump to S** if result is not zero.	2 or 4	2 or 13...20	2 or 4	2 or 13...28	D		
174	.	Branch S - Test	EEEE 1011100 10I DDDDDDDDD SSSSSSSSS	TJZ	D,{#}S			Test D and jump to S** if D is zero.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
175	.	Branch S - Test	EEEE 1011100 11I DDDDDDDDD SSSSSSSSS	TJNZ	D,{#}S			Test D and jump to S** if D is not zero.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
176	.	Branch S - Test	EEEE 1011101 00I DDDDDDDDD SSSSSSSSS	TJF	D,{#}S			Test D and jump to S** if D is full (D = $FFFF_FFFF).	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
177	.	Branch S - Test	EEEE 1011101 01I DDDDDDDDD SSSSSSSSS	TJNF	D,{#}S			Test D and jump to S** if D is not full (D != $FFFF_FFFF).	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
178	.	Branch S - Test	EEEE 1011101 10I DDDDDDDDD SSSSSSSSS	TJS	D,{#}S			Test D and jump to S** if D is signed (D[31] = 1).	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
179	.	Branch S - Test	EEEE 1011101 11I DDDDDDDDD SSSSSSSSS	TJNS	D,{#}S			Test D and jump to S** if D is not signed (D[31] = 0).	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
180	.	Branch S - Test	EEEE 1011110 00I DDDDDDDDD SSSSSSSSS	TJV	D,{#}S			Test D and jump to S** if D overflowed (D[31] != C, C = 'correct sign' from last addition/subtraction).	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
181	.	Events - Branch	EEEE 1011110 01I 000000000 SSSSSSSSS	JINT	{#}S			Jump to S** if INT event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
182	.	Events - Branch	EEEE 1011110 01I 000000001 SSSSSSSSS	JCT1	{#}S			Jump to S** if CT1 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
183	.	Events - Branch	EEEE 1011110 01I 000000010 SSSSSSSSS	JCT2	{#}S			Jump to S** if CT2 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
184	.	Events - Branch	EEEE 1011110 01I 000000011 SSSSSSSSS	JCT3	{#}S			Jump to S** if CT3 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
185	.	Events - Branch	EEEE 1011110 01I 000000100 SSSSSSSSS	JSE1	{#}S			Jump to S** if SE1 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
186	.	Events - Branch	EEEE 1011110 01I 000000101 SSSSSSSSS	JSE2	{#}S			Jump to S** if SE2 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
187	.	Events - Branch	EEEE 1011110 01I 000000110 SSSSSSSSS	JSE3	{#}S			Jump to S** if SE3 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
188	.	Events - Branch	EEEE 1011110 01I 000000111 SSSSSSSSS	JSE4	{#}S			Jump to S** if SE4 event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
189	.	Events - Branch	EEEE 1011110 01I 000001000 SSSSSSSSS	JPAT	{#}S			Jump to S** if PAT event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
190	.	Events - Branch	EEEE 1011110 01I 000001001 SSSSSSSSS	JFBW	{#}S			Jump to S** if FBW event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
191	.	Events - Branch	EEEE 1011110 01I 000001010 SSSSSSSSS	JXMT	{#}S			Jump to S** if XMT event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
192	.	Events - Branch	EEEE 1011110 01I 000001011 SSSSSSSSS	JXFI	{#}S			Jump to S** if XFI event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
193	.	Events - Branch	EEEE 1011110 01I 000001100 SSSSSSSSS	JXRO	{#}S			Jump to S** if XRO event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
194	.	Events - Branch	EEEE 1011110 01I 000001101 SSSSSSSSS	JXRL	{#}S			Jump to S** if XRL event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
195	.	Events - Branch	EEEE 1011110 01I 000001110 SSSSSSSSS	JATN	{#}S			Jump to S** if ATN event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
196	.	Events - Branch	EEEE 1011110 01I 000001111 SSSSSSSSS	JQMT	{#}S			Jump to S** if QMT event flag is set.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
197	.	Events - Branch	EEEE 1011110 01I 000010000 SSSSSSSSS	JNINT	{#}S			Jump to S** if INT event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
198	.	Events - Branch	EEEE 1011110 01I 000010001 SSSSSSSSS	JNCT1	{#}S			Jump to S** if CT1 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
199	.	Events - Branch	EEEE 1011110 01I 000010010 SSSSSSSSS	JNCT2	{#}S			Jump to S** if CT2 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
200	.	Events - Branch	EEEE 1011110 01I 000010011 SSSSSSSSS	JNCT3	{#}S			Jump to S** if CT3 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
201	.	Events - Branch	EEEE 1011110 01I 000010100 SSSSSSSSS	JNSE1	{#}S			Jump to S** if SE1 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
202	.	Events - Branch	EEEE 1011110 01I 000010101 SSSSSSSSS	JNSE2	{#}S			Jump to S** if SE2 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
203	.	Events - Branch	EEEE 1011110 01I 000010110 SSSSSSSSS	JNSE3	{#}S			Jump to S** if SE3 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
204	.	Events - Branch	EEEE 1011110 01I 000010111 SSSSSSSSS	JNSE4	{#}S			Jump to S** if SE4 event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
205	.	Events - Branch	EEEE 1011110 01I 000011000 SSSSSSSSS	JNPAT	{#}S			Jump to S** if PAT event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
206	.	Events - Branch	EEEE 1011110 01I 000011001 SSSSSSSSS	JNFBW	{#}S			Jump to S** if FBW event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
207	.	Events - Branch	EEEE 1011110 01I 000011010 SSSSSSSSS	JNXMT	{#}S			Jump to S** if XMT event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
208	.	Events - Branch	EEEE 1011110 01I 000011011 SSSSSSSSS	JNXFI	{#}S			Jump to S** if XFI event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
209	.	Events - Branch	EEEE 1011110 01I 000011100 SSSSSSSSS	JNXRO	{#}S			Jump to S** if XRO event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			
210	.	Events - Branch	EEEE 1011110 01I 000011101 SSSSSSSSS	JNXRL	{#}S			Jump to S** if XRL event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28			

211	.	Events - Branch	EEEE 1011110 01I 000011110 SSSSSSSS	JNATN	{#}S			Jump to S** if ATN event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28				
212	.	Events - Branch	EEEE 1011110 01I 000011111 SSSSSSSS	JNQMT	{#}S			Jump to S** if QMT event flag is clear.	2 or 4	2 or 13...20	2 or 4	2 or 13...28				
213		Miscellaneous	EEEE 1011110 1LI DDDDDDDD SSSSSSSS	<empty>	{#}D,{#}S			<empty>								
214	.	Miscellaneous	EEEE 1011111 0LI DDDDDDDD SSSSSSSS	<empty>	{#}D,{#}S			<empty>								
215	.	Events - Configuratio	EEEE 1011111 1LI DDDDDDDD SSSSSSSS	SETPAT	{#}D,{#}S			Set pin pattern for PAT event. C selects INA/INB, Z selects =/!=, D provides mask value, S provides match value.	2	same	2	same				
216	.	Smart Pins	EEEE 1100000 0LI DDDDDDDD SSSSSSSS	WRPIN	{#}D,{#}S			Write D to mode register of smart pin S[5:0], acknowledge smart pin.	2	same	2	same				
217	alias	Smart Pins	EEEE 1100000 01I 000000001 SSSSSSSS	AKPIN	{#}S			Acknowledge smart pin S[5:0].	2	same	2	same				
218	.	Smart Pins	EEEE 1100000 1LI DDDDDDDD SSSSSSSS	WXPIN	{#}D,{#}S			Write D to parameter "X" of smart pin S[5:0], acknowledge smart pin.	2	same	2	same				
219	.	Smart Pins	EEEE 1100001 0LI DDDDDDDD SSSSSSSS	WYPIN	{#}D,{#}S			Write D to parameter "Y" of smart pin S[5:0], acknowledge smart pin.	2	same	2	same				
220	.	Lookup Table	EEEE 1100001 1LI DDDDDDDD SSSSSSSS	WRLUT	{#}D,{#}S			Write D to LUT address S[8:0].	2	same	2	same				
221	.	Hub RAM - Write	EEEE 1100010 0LI DDDDDDDD SSSSSSSS	WRBYTE	{#}D,{#}S/P			Write byte in D[7:0] to hub address {#}S/PTRx.	3...10	3...20	3...18	3...38				Write
222	.	Hub RAM - Write	EEEE 1100010 1LI DDDDDDDD SSSSSSSS	WRWORD	{#}D,{#}S/P			Write word in D[15:0] to hub address {#}S/PTRx.	3...10*	3...20 *	3...18 *	3...38 *				Write
223	.	Hub RAM - Write	EEEE 1100011 0LI DDDDDDDD SSSSSSSS	WRLONG	{#}D,{#}S/P			Write long in D[31:0] to hub address {#}S/PTRx. Prior SETQ/SETQ2 invokes cog/LUT block transfer.	3...10*	3...20 *	3...18 *	3...38 *				Write
224	alias	Hub RAM - Write	EEEE 1100011 0LI DDDDDDDD 101100001	PUSHA	{#}D			Write long in D[31:0] to hub address PTRA++.	3...10*	3...20 *	3...18 *	3...38 *				Write
225	alias	Hub RAM - Write	EEEE 1100011 0LI DDDDDDDD 111100001	PUSHB	{#}D			Write long in D[31:0] to hub address PTRB++.	3...10*	3...20 *	3...18 *	3...38 *				Write
226	.	Hub FIFO - New Read	EEEE 1100011 1LI DDDDDDDD SSSSSSSS	RDFAST	{#}D,{#}S			Begin new fast hub read via FIFO. D[31] = no wait, D[13:0] = block size in 64-byte units (0 = max), S[19:0] = block start address.	or WRFAST finish + 10...!	FIFO IN USE	or WRFAST finish + 10...!	FIFO IN USE				
227	.	Hub FIFO - New Write	EEEE 1100100 0LI DDDDDDDD SSSSSSSS	WRFAST	{#}D,{#}S			Begin new fast hub write via FIFO. D[31] = no wait, D[13:0] = block size in 64-byte units (0 = max), S[19:0] = block start address.	2 or WRFAST finish + 3	FIFO IN USE	2 or WRFAST finish + 3	FIFO IN USE				
228	.	Hub FIFO - New Block	EEEE 1100100 1LI DDDDDDDD SSSSSSSS	FBLOCK	{#}D,{#}S			Set next block for when block wraps. D[13:0] = block size in 64-byte units (0 = max), S[19:0] = block start address.	2	FIFO IN USE	2	FIFO IN USE				
229	.	Streamer	EEEE 1100101 0LI DDDDDDDD SSSSSSSS	XINIT	{#}D,{#}S			Issue streamer command immediately, zeroing phase.	2	same	2	same				
230	alias	Streamer	EEEE 1100101 011 000000000 000000000	XSTOP				Stop streamer immediately.	2	same	2	same				
231	.	Streamer	EEEE 1100101 1LI DDDDDDDD SSSSSSSS	XZERO	{#}D,{#}S			Buffer new streamer command to be issued on final NCO rollover of current command, zeroing phase.	2+	same	2+	same				
232	.	Streamer	EEEE 1100110 0LI DDDDDDDD SSSSSSSS	XCONT	{#}D,{#}S			Buffer new streamer command to be issued on final NCO rollover of current command, continuing phase.	2+	same	2+	same				
233	.	Branch Repeat	EEEE 1100110 1LI DDDDDDDD SSSSSSSS	REP	{#}D,{#}S			Execute next D[8:0] instructions S times. If S = 0, repeat infinitely. If D[8:0] = 0, nothing repeats.	2	same	2	same				
234	.	Hub Control - Cogs	EEEE 1100111 CLI DDDDDDDD SSSSSSSS	COGINIT	{#}D,{#}S	{WC}		Start cog selected by D. S[19:0] sets hub startup address and PTRB of cog. Prior SETQ sets PTRA of cog.	2...9, +2 if result	same	2...17, +2 if result	same	D if reg and WC			
235	.	CORDIC Solver	EEEE 1101000 0LI DDDDDDDD SSSSSSSS	QMUL	{#}D,{#}S			Begin CORDIC unsigned multiplication of D * S. GETQX/GETQY retrieves lower/upper product.	2...9	same	2...17	same				
236	.	CORDIC Solver	EEEE 1101000 1LI DDDDDDDD SSSSSSSS	QDIV	{#}D,{#}S			Begin CORDIC unsigned division of {SETQ value or 32'b0, D} / S. GETQX/GETQY retrieves quotient/remainder.	2...9	same	2...17	same				
237	.	CORDIC Solver	EEEE 1101001 0LI DDDDDDDD SSSSSSSS	QFRAC	{#}D,{#}S			Begin CORDIC unsigned division of {D, SETQ value or 32'b0} / S. GETQX/GETQY retrieves quotient/remainder.	2...9	same	2...17	same				
238	.	CORDIC Solver	EEEE 1101001 1LI DDDDDDDD SSSSSSSS	QSQRT	{#}D,{#}S			Begin CORDIC square root of {S, D}. GETQX retrieves root.	2...9	same	2...17	same				
239	.	CORDIC Solver	EEEE 1101010 0LI DDDDDDDD SSSSSSSS	QROTATE	{#}D,{#}S			Begin CORDIC rotation of point (D, SETQ value or 32'b0) by angle S. GETQX/GETQY retrieves X/Y.	2...9	same	2...17	same				
240	.	CORDIC Solver	EEEE 1101010 1LI DDDDDDDD SSSSSSSS	QVECTOR	{#}D,{#}S			Begin CORDIC vectoring of point (D, S). GETQX/GETQY retrieves length/angle.	2...9	same	2...17	same				
241	.	Hub Control - Multi	EEEE 1101011 00L DDDDDDDD 000000000	HUBSET	{#}D			Set hub configuration to D.	2...9	same	2...17	same				
242	.	Hub Control - Cogs	EEEE 1101011 C0L DDDDDDDD 000000001	COGID	{#}D	{WC}		If D is register and no WC, get cog ID (0 to 15) into D. If WC, check status of cog D[3:0], C = 1 if on.	2...9, +2 if result	same	2...17, +2 if result	same	D if reg and !WC			
243	.	Hub Control - Cogs	EEEE 1101011 00L DDDDDDDD 000000011	COGSTOP	{#}D			Stop cog D[3:0].	2...9	same	2...17	same				
244	.	Hub Control - Locks	EEEE 1101011 C00 DDDDDDDD 000000100	LOCKNEW	D	{WC}		Request a LOCK. D will be written with the LOCK number (0 to 15). C = 1 if no LOCK available.	4...11	same	4...19	same	D			
245	.	Hub Control - Locks	EEEE 1101011 00L DDDDDDDD 000000101	LOCKRET	{#}D			Return LOCK D[3:0] for reallocation.	2...9	same	2...17	same				
246	.	Hub Control - Locks	EEEE 1101011 C0L DDDDDDDD 000000110	LOCKTRY	{#}D	{WC}		Try to get LOCK D[3:0]. C = 1 if got LOCK. LOCKREL releases LOCK. LOCK is also released if owner cog stops or restarts.	2...9, +2 if result	same	2...17, +2 if result	same				
247	.	Hub Control - Locks	EEEE 1101011 C0L DDDDDDDD 000000111	LOCKREL	{#}D	{WC}		Release LOCK D[3:0]. If D is a register and WC, get current/last cog id of LOCK owner into D and LOCK status into C.	2...9, +2 if result	same	2...17, +2 if result	same				
248	.	CORDIC Solver	EEEE 1101011 00L DDDDDDDD 000001110	QLOG	{#}D			Begin CORDIC number-to-logarithm conversion of D. GETQX retrieves log {5'whole_exponent, 27'fractional_exponent}.	2...9	same	2...17	same				
249	.	CORDIC Solver	EEEE 1101011 00L DDDDDDDD 000001111	QEXP	{#}D			Begin CORDIC logarithm-to-number conversion of D. GETQX retrieves number.	2...9	same	2...17	same				
250	.	Hub FIFO - Read	EEEE 1101011 C20 DDDDDDDD 000010000	RFBYTE	D	{WC/WZ/WCZ}		Used after RDFAST. Read zero-extended byte from FIFO into D. C = MSB of byte. *	2	FIFO IN USE	2	FIFO IN USE	D			
251	.	Hub FIFO - Read	EEEE 1101011 C20 DDDDDDDD 000010001	RFWORD	D	{WC/WZ/WCZ}		Used after RDFAST. Read zero-extended word from FIFO into D. C = MSB of word. *	2	FIFO IN USE	2	FIFO IN USE	D			
252	.	Hub FIFO - Read	EEEE 1101011 C20 DDDDDDDD 000010010	RFLONG	D	{WC/WZ/WCZ}		Used after RDFAST. Read long from FIFO into D. C = MSB of long. *	2	FIFO IN USE	2	FIFO IN USE	D			
253	.	Hub FIFO - Read	EEEE 1101011 C20 DDDDDDDD 000010011	RFVAR	D	{WC/WZ/WCZ}		Used after RDFAST. Read zero-extended 1..4-byte value from FIFO into D. C = 0. *	2	FIFO IN USE	2	FIFO IN USE	D			
254	.	Hub FIFO - Read	EEEE 1101011 C20 DDDDDDDD 000010100	RFVARS	D	{WC/WZ/WCZ}		Used after RDFAST. Read sign-extended 1..4-byte value from FIFO into D. C = MSB of value. *	2	FIFO IN USE	2	FIFO IN USE	D			
255	.	Hub FIFO - Write	EEEE 1101011 00L DDDDDDDD 000010101	WFBYTE	{#}D			Used after WRFAST. Write byte in D[7:0] into FIFO.	2	FIFO IN USE	2	FIFO IN USE				
256	.	Hub FIFO - Write	EEEE 1101011 00L DDDDDDDD 000010110	WFWORD	{#}D			Used after WRFAST. Write word in D[15:0] into FIFO.	2	FIFO IN USE	2	FIFO IN USE				
257	.	Hub FIFO - Write	EEEE 1101011 00L DDDDDDDD 000010111	WFLONG	{#}D			Used after WRFAST. Write long in D[31:0] into FIFO.	2	FIFO IN USE	2	FIFO IN USE				
258	.	CORDIC Solver	EEEE 1101011 C20 DDDDDDDD 000011000	GETQX	D	{WC/WZ/WCZ}		Retrieve CORDIC result X into D. Waits, in case result not ready. C = X[31]. *	2...58	same	2...58	same	D			
259	.	CORDIC Solver	EEEE 1101011 C20 DDDDDDDD 000011001	GETQY	D	{WC/WZ/WCZ}		Retrieve CORDIC result Y into D. Waits, in case result not ready. C = Y[31]. *	2...58	same	2...58	same	D			
260	.	Miscellaneous	EEEE 1101011 000 DDDDDDDD 000011010	GETCT	D			Get CT into D. CT (32 bits) is cleared on reset and then increments on every clock.	2	same	2	same	D			
261	.	Miscellaneous	EEEE 1101011 C20 DDDDDDDD 000011011	GETRND	D	{WC/WZ/WCZ}		Get RND into D/C/Z. RND is the PRNG that updates on every clock. D = RND[31:0], C = RND[31], Z = RND[30], unique per cog.	2	same	2	same	D			
262	alias	Miscellaneous	EEEE 1101011 CZ1 000000000 000011011	GETRND		WC/WZ/WCZ		Get RND into C/Z. C = RND[31], Z = RND[30], unique per cog.	2	same	2	same				
263	.	Smart Pins	EEEE 1101011 00L DDDDDDDD 000011100	SETDACS	{#}D			DAC3 = D[31:24], DAC2 = D[23:16], DAC1 = D[15:8], DAC0 = D[7:0].	2	same	2	same				
264	.	Streamer	EEEE 1101011 00L DDDDDDDD 000011101	SETXFRQ	{#}D			Set streamer NCO frequency to D.	2	same	2	same				
265	.	Streamer	EEEE 1101011 000 DDDDDDDD 000011110	GETXACC	D			Get the streamer's Goertzel X accumulator into D and the Y accumulator into the next instruction's S, clear accumulators.	2	same	2	same	D			
266	.	Miscellaneous	EEEE 1101011 00L DDDDDDDD 000011111	WAITX	{#}D	{WC/WZ/WCZ}		Wait 2 + D clocks if no WC/WZ/WCZ. If WC/WZ/WCZ, wait 2 + (D & RND) clocks. C/Z = 0.	2 + D	same	2 + D	same				
267	.	Events - Configuratio	EEEE 1101011 00L DDDDDDDD 000100000	SETSE1	{#}D			Set SE1 event configuration to D[8:0].	2	same	2	same				
268	.	Events - Configuratio	EEEE 1101011 00L DDDDDDDD 000100001	SETSE2	{#}D			Set SE2 event configuration to D[8:0].	2	same	2	same				
269	.	Events - Configuratio	EEEE 1101011 00L DDDDDDDD 000100010	SETSE3	{#}D			Set SE3 event configuration to D[8:0].	2	same	2	same				
270	.	Events - Configuratio	EEEE 1101011 00L DDDDDDDD 000100011	SETSE4	{#}D			Set SE4 event configuration to D[8:0].	2	same	2	same				
271	.	Events - Poll	EEEE 1101011 C20 000000000 000100100	POLLINT		{WC/WZ/WCZ}		Get INT event flag into C/Z, then clear it.	2	same	2	same				
272	.	Events - Poll	EEEE 1101011 C20 000000001 000100100	POLLCT1		{WC/WZ/WCZ}		Get CT1 event flag into C/Z, then clear it.	2	same	2	same				
273	.	Events - Poll	EEEE 1101011 C20 000000010 000100100	POLLCT2		{WC/WZ/WCZ}		Get CT2 event flag into C/Z, then clear it.	2	same	2	same				
274	.	Events - Poll	EEEE 1101011 C20 000000011 000100100	POLLCT3		{WC/WZ/WCZ}		Get CT3 event flag into C/Z, then clear it.	2	same	2	same				
275	.	Events - Poll	EEEE 1101011 C20 000000100 000100100	POLLSE1		{WC/WZ/WCZ}		Get SE1 event flag into C/Z, then clear it.	2	same	2	same				
276	.	Events - Poll	EEEE 1101011 C20 000000101 000100100	POLLSE2		{WC/WZ/WCZ}		Get SE2 event flag into C/Z, then clear it.	2	same	2	same				
277	.	Events - Poll	EEEE 1101011 C20 000000110 000100100	POLLSE3		{WC/WZ/WCZ}		Get SE3 event flag into C/Z, then clear it.	2	same	2	same				
278	.	Events - Poll	EEEE 1101011 C20 000000111 000100100	POLLSE4		{WC/WZ/WCZ}		Get SE4 event flag into C/Z, then clear it.	2	same	2	same				
279	.	Events - Poll	EEEE 1101011 C20 000001000 000100100	POLLPAT		{WC/WZ/WCZ}		Get PAT event flag into C/Z, then clear it.	2	same	2	same				
280	.	Events - Poll	EEEE 1101011 C20 000001001 000100100	POLLFBW		{WC/WZ/WCZ}		Get FBW event flag into C/Z, then clear it.	2	same	2	same				
281	.	Events - Poll	EEEE 1101011 C20 000001010 000100100	POLLXMT		{WC/WZ/WCZ}		Get XMT event flag into C/Z, then clear it.	2	same	2	same				
282	.	Events - Poll	EEEE 1101011 C20 000001011 000100100	POLLXFI		{WC/WZ/WCZ}		Get XFI event flag into C/Z, then clear it.	2	same	2	same				
283	.	Events - Poll	EEEE 1101011 C20 000001100 000100100	POLLXRO		{WC/WZ/WCZ}		Get XRO event flag into C/Z, then clear it.	2	same	2	same				
284	.	Events - Poll	EEEE 1101011 C20 000001101 000100100	POLLXRL		{WC/WZ/WCZ}		Get XRL event flag into C/Z, then clear it.	2	same	2	same				
285	.	Events - Poll	EEEE 1101011 C20 000001110 000100100	POLLATN		{WC/WZ/WCZ}		Get ATN event flag into C/Z, then clear it.	2	same	2	same				
286	.	Events - Poll	EEEE 1101011 C20 000001111 000100100	POLLQMT		{WC/WZ/WCZ}		Get QMT event flag into C/Z, then clear it.	2	same	2	same				
287	.	Events - Wait	EEEE 1101011 C20 000010000 000100100	WAITINT		{WC/WZ/WCZ}		Wait for INT event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
288	.	Events - Wait	EEEE 1101011 C20 000010001 000100100	WAITCT1		{WC/WZ/WCZ}		Wait for CT1 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
289	.	Events - Wait	EEEE 1101011 C20 000010010 000100100	WAITCT2		{WC/WZ/WCZ}		Wait for CT2 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
290	.	Events - Wait	EEEE 1101011 C20 000010011 000100100	WAITCT3		{WC/WZ/WCZ}		Wait for CT3 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
291	.	Events - Wait	EEEE 1101011 C20 000010100 000100100	WAITSE1		{WC/WZ/WCZ}		Wait for SE1 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
292	.	Events - Wait	EEEE 1101011 C20 000010101 000100100	WAITSE2		{WC/WZ/WCZ}		Wait for SE2 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
293	.	Events - Wait	EEEE 1101011 C20 000010110 000100100	WAITSE3		{WC/WZ/WCZ}		Wait for SE3 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
294	.	Events - Wait	EEEE 1101011 C20 000010111 000100100	WAITSE4		{WC/WZ/WCZ}		Wait for SE4 event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
295	.	Events - Wait	EEEE 1101011 C20 000011000 000100100	WAITPAT		{WC/WZ/WCZ}		Wait for PAT event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
296	.	Events - Wait	EEEE 1101011 C20 000011001 000100100	WAITFBW		{WC/WZ/WCZ}		Wait for FBW event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
297	.	Events - Wait	EEEE 1101011 C20 000011010 000100100	WAITXMT		{WC/WZ/WCZ}		Wait for XMT event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
298	.	Events - Wait	EEEE 1101011 C20 000011011 000100100	WAITXFI		{WC/WZ/WCZ}		Wait for XFI event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
299	.	Events - Wait	EEEE 1101011 C20 000011100 000100100	WAITXRO		{WC/WZ/WCZ}		Wait for XRO event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
300	.	Events - Wait	EEEE 1101011 C20 000011101 000100100	WAITXRL		{WC/WZ/WCZ}		Wait for XRL event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
301	.	Events - Wait	EEEE 1101011 C20 000011110 000100100	WAITATN		{WC/WZ/WCZ}		Wait for ATN event flag, then clear it. Prior SETQ sets optional CT timeout value. C/Z = timeout.	2+	same	2+	same				
302	.	Interrupts	EEEE 1101011 000 000100000 000100100	ALLOWI				Allow interrupts (default).	2	same	2	same				
303	.	Interrupts	EEEE 1101011 000 000100001 000100100	STALLI				Stall Interrupts.	2	same	2	same				
304	.	Interrupts	EEEE 1101011 000 000100010 000100100	TRGINT1				Trigger INT1, regardless of STALLI mode.	2	same	2	same				
305	.	Interrupts	EEEE 1101011 000 000100011 000100100	TRGINT2				Trigger INT2, regardless of STALLI mode.	2	same	2	same				
306	.	Interrupts	EEEE 1101011 000 000100100 000100100	TRGINT3				Trigger INT3, regardless of STALLI mode.	2	same	2	same				
307	.	Interrupts	EEEE 1101011 000 000100101 000100100	NIXINT1				Cancel INT1.	2	same	2	same				
308	.	Interrupts	EEEE 1101011 000 000100110 000100100	NIXINT2				Cancel INT2.	2	same	2	same				
309	.	Interrupts	EEEE 1101011 000 000100111 000100100	NIXINT3				Cancel INT3.	2	same	2	same				
310	.	Interrupts	EEEE 1101011 00L DDDDDDDD 000100101	SETINT1	{#}D			Set INT1 source to D[3:0].	2	same	2	same				
311	.	Interrupts	EEEE 1101011 00L DDDDDDDD 000100110	SETINT2	{#}D			Set INT2 source to D[3:0].	2	same	2	same				
312	.	Interrupts	EEEE 1101011 00L DDDDDDDD 000100111	SETINT3	{#}D			Set INT3 source to D[3:0].	2	same	2	same				
313	.	Miscellaneous	EEEE 1101011 00L DDDDDDDD 000101000	SETQ	{#}D			Set Q to D. Use before RDLONG/WRLONG/WMLONG to set block transfer. Also used before MUXQ/COGINIT/QDIV/QFRAC/QROTATE/WAITxxx.	2	same	2	same				
314	.	Miscellaneous	EEEE 1101011 00L DDDDDDDD 000101001	SETQ2	{#}D			Set Q to D. Use before RDLONG/WRLONG/WMLONG to set LUT block transfer.	2	same	2	same				
315	.	Miscellaneous	EEEE 1101011 00L DDDDDDDD 000101010	PUSH	{#}D			Push D onto stack.	2	same	2	same				Push

316	.	Miscellaneous	EEEE 1101011 CZ0 DDDDDDDDD 000101011	POP	D	{WC/WZ/WCZ}	Pop stack (K). C = K[31], Z = K[30], D = K.		2	same	2	same	D		Pop
317	.	Branch D - Jump	EEEE 1101011 CZ0 DDDDDDDDD 000101100	JMP	D	{WC/WZ/WCZ}	Jump to D.	C = D[31], Z = D[30], PC = D[19:0].	4	13...20	4	13...28			
318	.	Branch D - Call	EEEE 1101011 CZ0 DDDDDDDDD 000101101	CALL	D	{WC/WZ/WCZ}	Call to D by pushing {C, Z, 10'b0, PC[19:0]} onto stack.	C = D[31], Z = D[30], PC = D[19:0].	4	13...20	4	13...28			Push
319	.	Branch Return	EEEE 1101011 CZ1 000000000 000101101	RET		{WC/WZ/WCZ}	Return by popping stack (K).	C = K[31], Z = K[30], PC = K[19:0].	4	13...20	4	13...28			Pop
320	.	Branch D - Call	EEEE 1101011 CZ0 DDDDDDDDD 000101110	CALLA	D	{WC/WZ/WCZ}	Call to D by writing {C, Z, 10'b0, PC[19:0]} to hub long at PTRA++.	C = D[31], Z = D[30], PC = D[19:0].	5...12 *	14...32 *	5...20 *	14...64 *		Write	
321	.	Branch Return	EEEE 1101011 CZ1 000000000 000101110	RETA		{WC/WZ/WCZ}	Return by reading hub long (L) at --PTRA.	C = L[31], Z = L[30], PC = L[19:0].	11...18 *	20...40 *	11...26 *	20...70 *		Read	
322	.	Branch D - Call	EEEE 1101011 CZ0 DDDDDDDDD 000101111	CALLB	D	{WC/WZ/WCZ}	Call to D by writing {C, Z, 10'b0, PC[19:0]} to hub long at PTRB++.	C = D[31], Z = D[30], PC = D[19:0].	5...12 *	14...32 *	5...20 *	14...64 *		Write	
323	.	Branch Return	EEEE 1101011 CZ1 000000000 000101111	RETB		{WC/WZ/WCZ}	Return by reading hub long (L) at --PTRB.	C = L[31], Z = L[30], PC = L[19:0].	11...18 *	20...40 *	11...26 *	20...70 *		Read	
324	.	Branch D - Jump	EEEE 1101011 00L DDDDDDDDD 000110000	JMPREL	{#}D		Jump ahead/back by D instructions. For cogex, PC += D[19:0]. For hubex, PC += D[17:0] << 2.		4	13...20	4	13...28			
325	.	Branch D - Skip	EEEE 1101011 00L DDDDDDDDD 000110001	SKIP	{#}D		Skip instructions per D. Subsequent instructions 0..31 get cancelled for each '1' bit in D[0]..D[31].		2	same	2	same			
326	.	Branch D - Jump+Sk	EEEE 1101011 00L DDDDDDDDD 000110010	SKIPF	{#}D		Skip cog/LUT instructions fast per D. Like SKIP, but instead of cancelling instructions, the PC leaps over them.		2	ILLEGAL	2	ILLEGAL			
327	.	Branch D - Call+Skip	EEEE 1101011 00L DDDDDDDDD 000110011	EXECF	{#}D		Jump to D[9:0] in cog/LUT and set SKIPF pattern to D[31:10]. PC = {10'b0, D[9:0]}.		4	same	4	same			
328	.	Hub FIFO	EEEE 1101011 000 DDDDDDDDD 000110100	GETPTR	D		Get current FIFO hub pointer into D.		2	FIFO IN USE	2	FIFO IN USE	D		
329	.	Interrupts	EEEE 1101011 CZ0 DDDDDDDDD 000110101	GETBRK	D	WC/WZ/WCZ	Get breakpoint status into D according to WC/WZ/WCZ. C = 0. Z = 0.		2	same	2	same	D		
330	.	Interrupts	EEEE 1101011 00L DDDDDDDDD 000110101	COGBRK	{#}D		If in debug ISR, trigger asynchronous breakpoint in cog D[3:0]. Cog D[3:0] must have asynchronous breakpoint enabled.		2	same	2	same			
331	.	Interrupts	EEEE 1101011 00L DDDDDDDDD 000110110	BRK	{#}D		If in debug ISR, set next break condition to D. Else, trigger break if enabled, conditionally write break code to D[7:0].		2	same	2	same			
332	.	Lookup Table	EEEE 1101011 00L DDDDDDDDD 000110111	SETLUTS	{#}D		If D[0] = 1 then enable LUT sharing, where LUT writes within the adjacent odd/even companion cog are copied to this LUT.		2	same	2	same			
333	alias	Lookup Table	EEEE 1101011 001 000000000 000110111	LUTSOFF			Disable LUT sharing.		2	same	2	same			
334	alias	Lookup Table	EEEE 1101011 001 000000001 000110111	LUTSON			Enable LUT sharing.		2	same	2	same			
335	.	Color Space Converte	EEEE 1101011 00L DDDDDDDDD 000111000	SETCY	{#}D		Set the colorspace converter "CY" parameter to D[31:0].		2	same	2	same			
336	.	Color Space Converte	EEEE 1101011 00L DDDDDDDDD 000111001	SETCI	{#}D		Set the colorspace converter "CI" parameter to D[31:0].		2	same	2	same			
337	.	Color Space Converte	EEEE 1101011 00L DDDDDDDDD 000111010	SETCQ	{#}D		Set the colorspace converter "CQ" parameter to D[31:0].		2	same	2	same			
338	.	Color Space Converte	EEEE 1101011 00L DDDDDDDDD 000111011	SETCFRQ	{#}D		Set the colorspace converter "CFRQ" parameter to D[31:0].		2	same	2	same			
339	.	Color Space Converte	EEEE 1101011 00L DDDDDDDDD 000111100	SETCMOD	{#}D		Set the colorspace converter "CMOD" parameter to D[6:0].		2	same	2	same			
340	.	Pixel Mixer	EEEE 1101011 00L DDDDDDDDD 000111101	SETPIV	{#}D		Set BLNPIX/MIXPIX blend factor to D[7:0].		2	same	2	same			
341	.	Pixel Mixer	EEEE 1101011 00L DDDDDDDDD 000111110	SETPIX	{#}D		Set MIXPIX mode to D[5:0].		2	same	2	same			
342	.	Events - Attention	EEEE 1101011 00L DDDDDDDDD 000111111	COGATN	{#}D		Strobe "attention" of all cogs whose corresponging bits are high in D[15:0].		2	same	2	same			
343	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000000	TESTP	{#}D	WC/WZ	Test IN bit of pin D[5:0], write to C/Z. C/Z = IN[D[5:0]].		2	same	2	same			
344	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000001	TESTPN	{#}D	WC/WZ	Test !IN bit of pin D[5:0], write to C/Z. C/Z = !IN[D[5:0]].		2	same	2	same			
345	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000010	TESTP	{#}D	ANDC/ANDZ	Test IN bit of pin D[5:0], AND into C/Z. C/Z = C/Z AND IN[D[5:0]].		2	same	2	same			
346	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000011	TESTPN	{#}D	ANDC/ANDZ	Test !IN bit of pin D[5:0], AND into C/Z. C/Z = C/Z AND !IN[D[5:0]].		2	same	2	same			
347	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000100	TESTP	{#}D	ORC/ORZ	Test IN bit of pin D[5:0], OR into C/Z. C/Z = C/Z OR IN[D[5:0]].		2	same	2	same			
348	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000101	TESTPN	{#}D	ORC/ORZ	Test !IN bit of pin D[5:0], OR into C/Z. C/Z = C/Z OR !IN[D[5:0]].		2	same	2	same			
349	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000110	TESTP	{#}D	XORC/XORZ	Test IN bit of pin D[5:0], XOR into C/Z. C/Z = C/Z XOR IN[D[5:0]].		2	same	2	same			
350	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000111	TESTPN	{#}D	XORC/XORZ	Test !IN bit of pin D[5:0], XOR into C/Z. C/Z = C/Z XOR !IN[D[5:0]].		2	same	2	same			
351	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000000	DIRL	{#}D	{WCZ}	DIR bit of pin D[5:0] = 0.	C,Z = DIR bit.	2	same	2	same	DIRx		
352	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000001	DIRH	{#}D	{WCZ}	DIR bit of pin D[5:0] = 1.	C,Z = DIR bit.	2	same	2	same	DIRx		
353	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000010	DIRC	{#}D	{WCZ}	DIR bit of pin D[5:0] = C.	C,Z = DIR bit.	2	same	2	same	DIRx		
354	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000011	DIRNC	{#}D	{WCZ}	DIR bit of pin D[5:0] = !C.	C,Z = DIR bit.	2	same	2	same	DIRx		
355	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000100	DIRZ	{#}D	{WCZ}	DIR bit of pin D[5:0] = Z.	C,Z = DIR bit.	2	same	2	same	DIRx		
356	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000101	DIRNZ	{#}D	{WCZ}	DIR bit of pin D[5:0] = !Z.	C,Z = DIR bit.	2	same	2	same	DIRx		
357	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000110	DIRRND	{#}D	{WCZ}	DIR bit of pin D[5:0] = RND.	C,Z = DIR bit.	2	same	2	same	DIRx		
358	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001000111	DIRNOT	{#}D	{WCZ}	DIR bit of pin D[5:0] = !bit.	C,Z = DIR bit.	2	same	2	same	DIRx		
359	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001000	OUTL	{#}D	{WCZ}	OUT bit of pin D[5:0] = 0.	C,Z = OUT bit.	2	same	2	same	OUTx		
360	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001001	OUTH	{#}D	{WCZ}	OUT bit of pin D[5:0] = 1.	C,Z = OUT bit.	2	same	2	same	OUTx		
361	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001010	OUTC	{#}D	{WCZ}	OUT bit of pin D[5:0] = C.	C,Z = OUT bit.	2	same	2	same	OUTx		
362	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001011	OUTNC	{#}D	{WCZ}	OUT bit of pin D[5:0] = !C.	C,Z = OUT bit.	2	same	2	same	OUTx		
363	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001100	OUTZ	{#}D	{WCZ}	OUT bit of pin D[5:0] = Z.	C,Z = OUT bit.	2	same	2	same	OUTx		
364	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001101	OUTNZ	{#}D	{WCZ}	OUT bit of pin D[5:0] = !Z.	C,Z = OUT bit.	2	same	2	same	OUTx		
365	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001110	OUTRND	{#}D	{WCZ}	OUT bit of pin D[5:0] = RND.	C,Z = OUT bit.	2	same	2	same	OUTx		
366	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001001111	OUTNOT	{#}D	{WCZ}	OUT bit of pin D[5:0] = !bit.	C,Z = OUT bit.	2	same	2	same	OUTx		
367	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010000	FLTL	{#}D	{WCZ}	OUT bit of pin D[5:0] = 0.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
368	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010001	FLTH	{#}D	{WCZ}	OUT bit of pin D[5:0] = 1.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
369	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010010	FLTC	{#}D	{WCZ}	OUT bit of pin D[5:0] = C.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
370	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010011	FLTNC	{#}D	{WCZ}	OUT bit of pin D[5:0] = !C.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
371	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010100	FLTZ	{#}D	{WCZ}	OUT bit of pin D[5:0] = Z.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
372	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010101	FLTNZ	{#}D	{WCZ}	OUT bit of pin D[5:0] = !Z.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
373	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010110	FLTRND	{#}D	{WCZ}	OUT bit of pin D[5:0] = RND.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
374	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001010111	FLTNOT	{#}D	{WCZ}	OUT bit of pin D[5:0] = !bit.	DIR bit = 0. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
375	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011000	DRVL	{#}D	{WCZ}	OUT bit of pin D[5:0] = 0.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
376	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011001	DRVH	{#}D	{WCZ}	OUT bit of pin D[5:0] = 1.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
377	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011010	DRVC	{#}D	{WCZ}	OUT bit of pin D[5:0] = C.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
378	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011011	DRVNC	{#}D	{WCZ}	OUT bit of pin D[5:0] = !C.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
379	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011100	DRVZ	{#}D	{WCZ}	OUT bit of pin D[5:0] = Z.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
380	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011101	DRVNZ	{#}D	{WCZ}	OUT bit of pin D[5:0] = !Z.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
381	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011110	DRVRND	{#}D	{WCZ}	OUT bit of pin D[5:0] = RND.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
382	.	Pins	EEEE 1101011 CZL DDDDDDDDD 001011111	DRVNOT	{#}D	{WCZ}	OUT bit of pin D[5:0] = !bit.	DIR bit = 1. C,Z = OUT bit.	2	same	2	same	DIRx* + OUTx		
383	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100000	SPLITB	D		Split every 4th bit of S into bytes of D. D = {S[31], S[27], S[23], S[19], ...S[12], S[8], S[4], S[0]}.		2	same	2	same	D		
384	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100001	MERGEB	D		Merge bits of bytes in S into D. D = {S[31], S[23], S[15], S[7], ...S[24], S[16], S[8], S[0]}.		2	same	2	same	D		
385	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100010	SPLITW	D		Split bits of S into words of D. D = {S[31], S[29], S[27], S[25], ...S[6], S[4], S[2], S[0]}.		2	same	2	same	D		
386	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100011	MERGEW	D		Merge bits of words in S into D. D = {S[31], S[15], S[30], S[14], ...S[17], S[1], S[16], S[0]}.		2	same	2	same	D		
387	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100100	SEUSSF	D		Relocate and periodically invert bits from S into D. Returns to original value on 32nd iteration. Forward pattern.		2	same	2	same	D		
388	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100101	SEUSSR	D		Relocate and periodically invert bits from S into D. Returns to original value on 32nd iteration. Reverse pattern.		2	same	2	same	D		
389	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100110	RGBSQZ	D		Squeeze 8:8:8 RGB value in S[31:8] into 5:6:5 value in D[15:0]. D = {15'b0, S[31:27], S[23:18], S[15:11]}.		2	same	2	same	D		
390	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001100111	RGBEXP	D		Expand 5:6:5 RGB value in S[15:0] into 8:8:8 value in D[31:8]. D = {S[15:11,15:13], S[10:5,10:9], S[4:0,4:2], 8'b0}.		2	same	2	same	D		
391	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001101000	XORO32	D		Iterate D with xoroshiro32+ PRNG algorithm and put PRNG result into next instruction's S.		2	same	2	same	D		
392	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001101001	REV	D		Reverse D bits. D = D[0:31].		2	same	2	same	D		
393	.	Math and Logic	EEEE 1101011 CZ0 DDDDDDDDD 001101010	RCZR	D	{WC/WZ/WCZ}	Rotate C,Z right through D. D = {C, Z, D[31:2]}. C = D[1], Z = D[0].		2	same	2	same	D		
394	.	Math and Logic	EEEE 1101011 CZ0 DDDDDDDDD 001101011	RCZL	D	{WC/WZ/WCZ}	Rotate C,Z left through D. D = {D[29:0], C, Z}. C = D[31], Z = D[30].		2	same	2	same	D		
395	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001101100	WRC	D		Write 0 or 1 to D, according to C. D = {31'b0, C}.		2	same	2	same	D		
396	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001101101	WRNC	D		Write 0 or 1 to D, according to !C. D = {31'b0, !C}.		2	same	2	same	D		
397	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001101110	WRZ	D		Write 0 or 1 to D, according to Z. D = {31'b0, Z}.		2	same	2	same	D		
398	.	Math and Logic	EEEE 1101011 000 DDDDDDDDD 001101111	WRNZ	D		Write 0 or 1 to D, according to !Z. D = {31'b0, !Z}.		2	same	2	same	D		
399	.	Math and Logic	EEEE 1101011 CZ1 0cccczzzz 001101111	MODCZ	c,z	{WC/WZ/WCZ}	Modify C and Z according to cccc and zzzz. C = cccc[{C,Z}], Z = zzzz[{C,Z}].		2	same	2	same			
400	alias	Math and Logic	EEEE 1101011 C01 0cccc0000 001101111	MODC	c	{WC}	Modify C according to cccc. C = cccc[{C,Z}].		2	same	2	same			
401	alias	Math and Logic	EEEE 1101011 0Z1 00000zzzz 001101111	MODZ	z	{WZ}	Modify Z according to zzzz. Z = zzzz[{C,Z}].		2	same	2	same			
402	.	Branch A - Jump	EEEE 1101100 RAA AAAAAAAAA AAAAAAAAA	JMP	#A		Jump to A.	If R = 1, PC += A, else PC = A.	4	13...20	4	13...28			
403	.	Branch A - Call	EEEE 1101101 RAA AAAAAAAAA AAAAAAAAA	CALL	#A		Call to A by pushing {C, Z, 10'b0, PC[19:0]} onto stack.	If R = 1, PC += A, else PC = A.	4	13...20	4	13...28			Push
404	.	Branch A - Call	EEEE 1101110 RAA AAAAAAAAA AAAAAAAAA	CALLA	#A		Call to A by writing {C, Z, 10'b0, PC[19:0]} to hub long at PTRA++.	If R = 1, PC += A, else PC = A.	5...12 *	14...32 *	5...20 *	14...64 *		Write	
405	.	Branch A - Call	EEEE 1101111 RAA AAAAAAAAA AAAAAAAAA	CALLB	#A		Call to A by writing {C, Z, 10'b0, PC[19:0]} to hub long at PTRB++.	If R = 1, PC += A, else PC = A.	5...12 *	14...32 *	5...20 *	14...64 *		Write	
406	.	Branch A - Call	EEEE 11100WW RAA AAAAAAAAA AAAAAAAAA	CALLD	PA/PB/PTRA/PTRB,#A		Call to A by writing {C, Z, 10'b0, PC[19:0]} to PA/PB/PTRA/PTRB (per W).	If R = 1, PC += A, else PC = A.	4	13...20	4	13...28	Per W		
407	.	Branch A - Call	EEEE 11101WW RAA AAAAAAAAA AAAAAAAAA	LOC	PA/PB/PTRA/PTRB,#A		Get {12'b0, address[19:0]} into PA/PB/PTRA/PTRB (per W). If R = 1, address = PC + A, else address = A.		2	same	2	same	Per W		
408	.	Miscellaneous	EEEE 11110NN NNN NNNNNNNNN NNNNNNNNN	AUGS	#N		Queue #N[31:9] to be used as upper 23 bits for next #S occurrence, so that the next 9-bit #S will be augmented to 32 bits.		2	same	2	same			
409	.	Miscellaneous	EEEE 11111NN NNN NNNNNNNNN NNNNNNNNN	AUGD	#N		Queue #N[31:9] to be used as upper 23 bits for next #D occurrence, so that the next 9-bit #D will be augmented to 32 bits.		2	same	2	same			
410															
411	.	Instruction Prefix	0000 ------- --- --------- ---------	_RET_	<inst>	<ops>	Execute <inst> always and return if no branch. If <inst> is not branching then return by popping stack[19:0] into PC.		+2	+11...18	+2	+11...26			Pop
412	.	Instruction Prefix	0001 ------- --- --------- ---------	IF_NC_AND_NZ	<inst>	<ops>	Execute <inst> if C = 0 and Z = 0.								
413	alias	Instruction Prefix	0001 ------- --- --------- ---------	IF_NZ_AND_NC	<inst>	<ops>	Execute <inst> if C = 0 and Z = 0.								
414	alias	Instruction Prefix	0001 ------- --- --------- ---------	IF_A	<inst>	<ops>	Execute <inst> if C = 0 and Z = 0, or if 'above' after a comparison/subtraction.								
415	alias	Instruction Prefix	0001 ------- --- --------- ---------	IF_00	<inst>	<ops>	Execute <inst> if C = 0 and Z = 0.								
416	.	Instruction Prefix	0010 ------- --- --------- ---------	IF_NC_AND_Z	<inst>	<ops>	Execute <inst> if C = 0 and Z = 1.								
417	alias	Instruction Prefix	0010 ------- --- --------- ---------	IF_Z_AND_NC	<inst>	<ops>	Execute <inst> if C = 0 and Z = 1.								
418	alias	Instruction Prefix	0010 ------- --- --------- ---------	IF_01	<inst>	<ops>	Execute <inst> if C = 0 and Z = 1.								
419	.	Instruction Prefix	0011 ------- --- --------- ---------	IF_NC	<inst>	<ops>	Execute <inst> if C = 0.								
420	alias	Instruction Prefix	0011 ------- --- --------- ---------	IF_AE	<inst>	<ops>	Execute <inst> if C = 0, or if 'above or equal' after a comparison/subtraction.								

421	alias	**Instruction Prefix**	0011	--------	---	---------	---------	**IF_0X**	**<inst>**	**<ops>**	Execute <inst> if C = 0.
422	.	**Instruction Prefix**	0100	--------	---	---------	---------	**IF_C_AND_NZ**	**<inst>**	**<ops>**	Execute <inst> if C = 1 and Z = 0.
423	alias	**Instruction Prefix**	0100	--------	---	---------	---------	**IF_NZ_AND_C**	**<inst>**	**<ops>**	Execute <inst> if C = 1 and Z = 0.
424	alias	**Instruction Prefix**	0100	--------	---	---------	---------	**IF_10**	**<inst>**	**<ops>**	Execute <inst> if C = 1 and Z = 0.
425	.	**Instruction Prefix**	0101	--------	---	---------	---------	**IF_NZ**	**<inst>**	**<ops>**	Execute <inst> if Z = 0.
426	alias	**Instruction Prefix**	0101	--------	---	---------	---------	**IF_NE**	**<inst>**	**<ops>**	Execute <inst> if Z = 0, or if 'not equal' after a comparison/subtraction.
427	alias	**Instruction Prefix**	0101	--------	---	---------	---------	**IF_X0**	**<inst>**	**<ops>**	Execute <inst> if Z = 0.
428	.	**Instruction Prefix**	0110	--------	---	---------	---------	**IF_C_NE_Z**	**<inst>**	**<ops>**	Execute <inst> if C != Z.
429	alias	**Instruction Prefix**	0110	--------	---	---------	---------	**IF_Z_NE_C**	**<inst>**	**<ops>**	Execute <inst> if C != Z.
430	alias	**Instruction Prefix**	0110	--------	---	---------	---------	**IF_DIFF**	**<inst>**	**<ops>**	Execute <inst> if C != Z.
431	.	**Instruction Prefix**	0111	--------	---	---------	---------	**IF_NC_OR_NZ**	**<inst>**	**<ops>**	Execute <inst> if C = 0 or Z = 0.
432	alias	**Instruction Prefix**	0111	--------	---	---------	---------	**IF_NZ_OR_NC**	**<inst>**	**<ops>**	Execute <inst> if C = 0 or Z = 0.
433	alias	**Instruction Prefix**	0111	--------	---	---------	---------	**IF_NOT_11**	**<inst>**	**<ops>**	Execute <inst> if C = 0 or Z = 0.
434	.	**Instruction Prefix**	1000	--------	---	---------	---------	**IF_C_AND_Z**	**<inst>**	**<ops>**	Execute <inst> if C = 1 and Z = 1.
435	alias	**Instruction Prefix**	1000	--------	---	---------	---------	**IF_Z_AND_C**	**<inst>**	**<ops>**	Execute <inst> if C = 1 and Z = 1.
436	alias	**Instruction Prefix**	1000	--------	---	---------	---------	**IF_11**	**<inst>**	**<ops>**	Execute <inst> if C = 1 and Z = 1.
437	.	**Instruction Prefix**	1001	--------	---	---------	---------	**IF_C_EQ_Z**	**<inst>**	**<ops>**	Execute <inst> if C = Z.
438	alias	**Instruction Prefix**	1001	--------	---	---------	---------	**IF_Z_EQ_C**	**<inst>**	**<ops>**	Execute <inst> if C = Z.
439	alias	**Instruction Prefix**	1001	--------	---	---------	---------	**IF_SAME**	**<inst>**	**<ops>**	Execute <inst> if C = Z.
440	.	**Instruction Prefix**	1010	--------	---	---------	---------	**IF_Z**	**<inst>**	**<ops>**	Execute <inst> if Z = 1.
441	alias	**Instruction Prefix**	1010	--------	---	---------	---------	**IF_E**	**<inst>**	**<ops>**	Execute <inst> if Z = 1, or if 'equal' after a comparison/subtraction.
442	alias	**Instruction Prefix**	1010	--------	---	---------	---------	**IF_X1**	**<inst>**	**<ops>**	Execute <inst> if Z = 1.
443	.	**Instruction Prefix**	1011	--------	---	---------	---------	**IF_NC_OR_Z**	**<inst>**	**<ops>**	Execute <inst> if C = 0 or Z = 1.
444	alias	**Instruction Prefix**	1011	--------	---	---------	---------	**IF_Z_OR_NC**	**<inst>**	**<ops>**	Execute <inst> if C = 0 or Z = 1.
445	alias	**Instruction Prefix**	1011	--------	---	---------	---------	**IF_NOT_10**	**<inst>**	**<ops>**	Execute <inst> if C = 0 or Z = 1.
446	.	**Instruction Prefix**	1100	--------	---	---------	---------	**IF_C**	**<inst>**	**<ops>**	Execute <inst> if C = 1.
447	alias	**Instruction Prefix**	1100	--------	---	---------	---------	**IF_B**	**<inst>**	**<ops>**	Execute <inst> if C = 1, or if 'below' after a comparison/subtraction.
448	alias	**Instruction Prefix**	1100	--------	---	---------	---------	**IF_1X**	**<inst>**	**<ops>**	Execute <inst> if C = 1.
449	.	**Instruction Prefix**	1101	--------	---	---------	---------	**IF_C_OR_NZ**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 0.
450	alias	**Instruction Prefix**	1101	--------	---	---------	---------	**IF_NZ_OR_C**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 0.
451	alias	**Instruction Prefix**	1101	--------	---	---------	---------	**IF_NOT_01**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 0.
452	.	**Instruction Prefix**	1110	--------	---	---------	---------	**IF_C_OR_Z**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 1.
453	alias	**Instruction Prefix**	1110	--------	---	---------	---------	**IF_Z_OR_C**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 1.
454	alias	**Instruction Prefix**	1110	--------	---	---------	---------	**IF_BE**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 1, or if 'below or equal' after a comparison/subtraction.
455	alias	**Instruction Prefix**	1110	--------	---	---------	---------	**IF_NOT_00**	**<inst>**	**<ops>**	Execute <inst> if C = 1 or Z = 1.
456	.	**Instruction Prefix**	1111	--------	---	---------	---------		**<inst>**	**<ops>**	Execute <inst> always. This is the default when no instruction prefix is expressed.