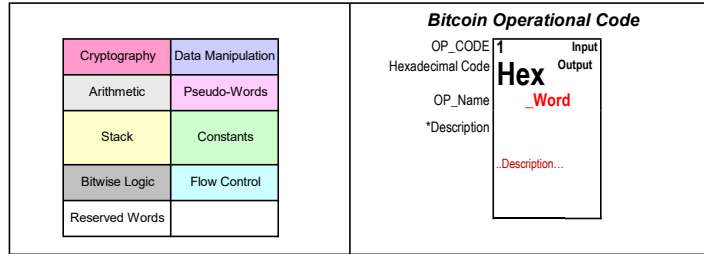


BTCOPIN OP CODES TABLE

BSV - Chronicle Upgrade

Copyright © BSV Colombia, J1 Pelaez, 2026



https://wiki.bitcoinsv.io/index.php/Opcodes_used_in_Bitcoin_Script

<https://docs.bsvblockchain.org/network-topology/nodes/sv-node/chronicle-release>

0 Nothing empty 0x00 _FALSE An empty array of bytes is pushed onto the stack.	126 x1 x2 0x7e _CAT Concatenates two strings.
166 hash 0xa6 _RIPEMD160 The input is hashed using RIPEMD-160.	167 hash 0xa7 _SHA1 The input is hashed using SHA-1.
168 hash 0xa8 _SHA256 The input is hashed using SHA-256.	169 hash 0xa9 _HASH160 The input is hashed twice: first with SHA-256 and then with RIPEMD-160.
170 hash 0xaa _HASH256 The input is hashed two times with SHA-256.	171 nothing 0xab _CODESEPARATOR All of the signature checking words will only match signatures to the data after the most
172 sig pubkey 0xac _CHECKSIG The entire transaction's outputs, inputs, and script (from the most recently-	173 sig pubkey 0xad _CHECKSIGVERIFY Same as OP_CHECKSIG, but OP_VERIFY is executed afterward.
174 x sig1 sig2... 0xae _CHECKMULTISIG Compares the first signature against each public key until it finds an ECDSA match.	175 x sig1 sig2... 0xaf _CHECKMULTISIGVER Same as OP_CHECKMULTISIG, but OP_VERIFY is executed afterward.
125 x1x2 0x7d _TUCK The item at the top of the stack is copied and inserted before the second-to-top item.	80 x2x1x2 0x50 _RESERVED
176 0xb0 _NOP1 The word is ignored. Does not mark transaction as invalid.	177 0xb1 _NOP2 The word is ignored. Does not mark transaction as invalid.
178 0xb2 _NOP3 The word is ignored. Does not mark transaction as invalid.	179 0xb3 _SUBSTR Returns substring defined by start index and length.
180 0xb4 _LEFT Produces a substring consisting only of the specified number of leftmost characters.	181 0xb5 _RIGHT Produces a substring consisting only of the specified number of rightmost
182 0xb6 _LSHIFTNUM Performs a numerical shift to left, preserving sign.	183 0xb7 _RSHIFTNUM Performs a numerical shift to right, preserving sign.
184 0xb8 _NOP9	185 0xb9 _NOP10
97 Nothing 0x61 _NOP Does nothing.	01-75 0x01-0x4d PUSHDATA BY Bytlength The next opcode bytes is data to be pushed onto stack Up to 75 bytes.
131 in 0x83 _INVERT Flips all of the bits in the input.	98 Nothing 0x62 _VER Puts the version of the protocol under which this transaction will be evaluated onto the stack.
132 x1 x2 0x84 _AND Boolean and between each bit in the inputs.	99 expression 0x63 _IF If the top stack value is TRUE, statement 1 is executed.
133 x1 x2 0x85 _OR Boolean or between each bit in the inputs.	100 expression 0x64 _NOTIF If the top stack value is FALSE, statement 1 is executed.
134 x1 x2 0x86 _XOR Boolean exclusive or between each bit in the inputs.	101 NA 0x65 _VERIFY Compares the tos with the executing transaction's version as an equivalence comparison as part
135 x1 x2 0x87 _EQUAL Returns 1 if the inputs are exactly equal, 0 otherwise.	102 NA 0x66 _VERNOTIF Compares the tos with the executing transaction's version as an equivalence comparison as part
136 x1 x2 0x88 _EQUALVERIFY Same as OP_EQUAL, but runs OP_VERIFY afterward.	103 expression 0x67 _ELSE If the preceding IF or NOTIF check was not valid then statement is executed.
137 0x89 _RESERVED1	104 expression 0x68 _ENDIF Ends an if/else block. All blocks must end, or the transaction is invalid.
138 0x8a _RESERVED2	105 true/false 0x69 _VERIFY Marks transaction as invalid if top stack value is not true. The top stack value is removed.
139 0x8b _1ADD 1 is added to the input.	253 0xfd _PUBKEYHASH Represents a public key hashed with OP_HASH160.
140 0x8c _1SUB 1 is subtracted from the input.	254 0xfe _PUBKEY Represents a public key compatible with OP_CHECKSIG.
141 0x8d _2MUL The input is multiplied by 2.	128 a b 0x80 _NUM2BIN Converts numeric value a into byte sequence of length b.
142 0x8e _2DIV The input is divided by 2.	129 x 0x81 _BIN2NUM Converts byte sequence x into a numeric value.
143 0x8f _NEGATIVE The sign of the input is flipped.	130 in 0x82 _SIZE Pushes the string length of the top element of the stack
144 0x90 _ABS The input is made positive.	106 Nothing 0x6a _RETURN OP_RETURN can also be used to create "False Return" outputs with a scriptPubKey consisting of
145 0x91 _NOT If the input is 0 or 1, it is flipped. Otherwise the output will be 0.	107 x1 0x6b _TOALTSTACK Puts the input onto the top of the alt stack. Removes it from the main stack.
146 0x92 _NOTEQUAL Returns 0 if the input is 0. 1 otherwise.	108 (alt)x1 0x6c _FROMALTSTACK Puts the input onto the top of the main stack. Removes it from the alt stack.
147 0x93 _ADD a is added to b.	109 x1 x2 0x6d _2DROP Removes the top two stack items.
148 a b 0x94 _SUB b is subtracted from a.	110 x1x2 0x6e _2DUP Duplicates the top two stack items.
149 a b 0x95 _MUL a is multiplied by b.	111 x1x2x3 0x6f _3DUP Duplicates the top three stack items.
150 a b 0x96 _DIV a is divided by b.	112 x1x2x3x4 0x70 _2OVER Copies the pair of items two spaces back in the stack to the front.
151 a b 0x97 _MOD Returns the remainder after dividing a by b.	113 x1x2x3x4x5x6 0x71 _2ROT The fifth and sixth items back are moved to the top of the stack.
152 a b 0x98 _LSHIFT Logical left shift b bits. Sign data is discarded.	114 x1x2x3x4 0x72 _2SWAP Swaps the top two pairs of items.
153 a b 0x99 _RSHIFT Logical right shift b bits. Sign data is discarded.	115 x 0x73 _IFDUP If the top stack value is not 0, duplicate it.
154 a b 0x9a _BOOLAND If both a and b are not 0, the output is 1. Otherwise 0.	116 Nothing 0x74 _DEPTH Counts the number of stack items onto the stack and places the value on the top
155 a b 0x9b _BOOLOR If a or b is not 0, the output is 1. Otherwise 0.	117 0x75 _DROP Removes the top stack item.
156 a b 0x9c _NUMEQUAL Returns 1 if the numbers are equal. 0 otherwise.	118 x 0x76 _DUP Duplicates the top stack item.
157 a b 0x9d _NUMEQUALVERIFY Same as OP_NUMEQUAL, but runs OP_VERIFY afterward.	119 x1x2 0x77 _NIP Removes the second-to-top stack item.
158 a b 0x9e _NUMNOTEQUAL Returns 1 if the numbers are not equal. 0 otherwise.	120 x1x2 0x78 _OVER Copies the second-to-top stack item to the top.
159 in 0x9f _LESSTHAN Returns 1 if a is less than b, 0 otherwise.	121 x1x2x1 0x79 _PICK The item n back in the stack is copied to the top.
160 in 0xa0 _GREATERTHAN Returns 1 if a is greater than b, 0 otherwise.	122 xn...x2x1x0 0x7a _ROLL The item n back in the stack is moved to the top.
161 in 0xa1 _LESSTHANOR Returns 1 if a is less than or equal to b, 0 otherwise.	123 x1x2x3 0x7b _ROT The top three items on the stack are rotated to the left.
162 in 0xa2 _GREATERTHANOR Returns 1 if a is greater than or equal to b, 0 otherwise.	124 x1x2 0x7c _SWAP The top two items on the stack are swapped.
163 in 0xa3 _MIN Returns the smaller of a and b.	
164 in 0xa4 _MAX Returns the larger of a and b.	
165 x min max 0xa5 _WITHIN Returns 1 if x is within the specified range (left-inclusive), 0 otherwise.	
166 x 0xa6 _RIPEMD160 The input is hashed using RIPEMD-160.	
167 hash 0xa7 _SHA1 The input is hashed using SHA-1.	
168 hash 0xa8 _SHA256 The input is hashed using SHA-256.	
169 hash 0xa9 _HASH160 The input is hashed twice: first with SHA-256 and then with RIPEMD-160.	
170 hash 0xaa _HASH256 The input is hashed two times with SHA-256.	
171 nothing 0xab _CODESEPARATOR All of the signature checking words will only match signatures to the data after the most	
172 sig pubkey 0xac _CHECKSIG The entire transaction's outputs, inputs, and script (from the most recently-	
173 sig pubkey 0xad _CHECKSIGVERIFY Same as OP_CHECKSIG, but OP_VERIFY is executed afterward.	
174 x sig1 sig2... 0xae _CHECKMULTISIG Compares the first signature against each public key until it finds an ECDSA match.	
175 x sig1 sig2... 0xaf _CHECKMULTISIGVER Same as OP_CHECKMULTISIG, but OP_VERIFY is executed afterward.	
176 0xb0 _NOP1 The word is ignored. Does not mark transaction as invalid.	
177 0xb1 _NOP2 The word is ignored. Does not mark transaction as invalid.	
178 0xb2 _NOP3 The word is ignored. Does not mark transaction as invalid.	
179 0xb3 _SUBSTR Returns substring defined by start index and length.	
180 0xb4 _LEFT Produces a substring consisting only of the specified number of leftmost characters.	
181 0xb5 _RIGHT Produces a substring consisting only of the specified number of rightmost	
182 0xb6 _LSHIFTNUM Performs a numerical shift to left, preserving sign.	
183 0xb7 _RSHIFTNUM Performs a numerical shift to right, preserving sign.	
184 0xb8 _NOP9	
185 0xb9 _NOP10	

Constants

Constants			82	Nothing	83	Nothing	84	Nothing	85	Nothing	86	Nothing	87	Nothing	88	Nothing	89	Nothing	90	Nothing	91	Nothing	9C	Nothing	93	Nothing	94	Nothing	95	Nothing	96	Nothing																						
			0x52	2	0x53	3	0x54	4	0x55	5	0x56	6	0x57	7	0x58	8	0x59	9	0x5A	10	0x5B	11	0x5C	12	0x5D	13	0x5E	14	0x5F	15	0x60	16																						
			<u>2</u>		<u>3</u>		<u>4</u>		<u>5</u>		<u>6</u>		<u>7</u>		<u>8</u>		<u>9</u>		<u>10</u>		<u>11</u>		<u>12</u>		<u>13</u>		<u>14</u>		<u>15</u>		<u>16</u>																							
			The number in the word name (2) is pushed onto the stack.		The number in the word name (3) is pushed onto the stack.		The number in the word name (4) is pushed onto the stack.		The number in the word name (5) is pushed onto the stack.		The number in the word name (6) is pushed onto the stack.		The number in the word name (7) is pushed onto the stack.		The number in the word name (8) is pushed onto the stack.		The number in the word name (9) is pushed onto the stack.		The number in the word name (10) is pushed onto the stack.		The number in the word name (11) is pushed onto the stack.		The number in the word name (12) is pushed onto the stack.		The number in the word name (13) is pushed onto the stack.		The number in the word name (14) is pushed onto the stack.		The number in the word name (15) is pushed onto the stack.		The number in the word name (16) is pushed onto the stack.																							
107	x1		108	(alt)x1		109	x1 x2		110	x1x2		111	x1x2x3		112	x1x2x3x4		113	x1x2x3x4x5x6		114	x1x2x3x4		115			116	Nothing		117			118	x	119	x1x2		120	x1x2		121	xn...x2x1x0		122	xn...x2x1x0		123	x1x2x3		124	x1x2			
0x6b	(alt)x1		0x6c	x1		0x6d	Nothing		0x6e	x1x2x1x2		0x6f	x1x2x3x1x2		0x70	x1x2x3x4		0x71	x3x4x5x6		0x72	x3x4x1x2		0x73	x/xx		0x74	stack size		0x75	Nothing		0x76	xx		0x77	x2		0x78	x1x2x1		0x79	xm...x2x1x0		0x7a	x2x3x1x0		0x7b	x2x3x1		0x7c	x2x1		
			<u>TOALTSTACK</u>		<u>2DROP</u>				<u>2DUP</u>		<u>3DUP</u>				<u>2OVER</u>				<u>2ROT</u>				<u>2SWAP</u>			<u>IFDUP</u>		<u>DEPTH</u>			<u>DROP</u>			<u>DUP</u>			<u>NIP</u>			<u>OVER</u>			<u>PICK</u>			<u>ROLL</u>			<u>ROT</u>			<u>SWAP</u>		
			Puts the input onto the top of the alt stack. Removes it from the main stack.		Removes the top two stack items.				Duplicates the top two stack items.		Duplicates the top three stack items.				Copies the pair of items two spaces back in the stack to the front.				The fifth and sixth items back are moved to the top of the stack.				Swaps the top two pairs of items.			If the top stack value is not 0, duplicate it.		Counts the number of stack items onto the stack and places the value on the top			Removes the top stack item.			Duplicates the top stack item.			Removes the second-to-top stack item.			Copies the second-to-top stack item to the top.			The item n back in the stack is copied to the top.			The item n back in the stack is moved to the top.			The top three items on the stack are rotated to the left.			The top two items on the stack are swapped.		

Stack