CASH CODES TABLE
BCH

Cryptography	Splice
Arithmetic	Pseudo-Words
Stack	Constants
Bitwise Logic	Flow Control
Reserved Words	Locktime
Introspection	

Operational Code

OP_CODE	1 — Input
Hexadecimal Code	**Hex** — Output
OP_Name	_Word
*Description	..Description...

https://reference.cash/protocol/blockchain/script

0 Nothing **0x00** empty _FALSE An empty array of bytes is pushed onto the stack.																	**126** x1 x2 **0x7e** Out _CAT Concatenates two strings.
166 in **0xa6** hash _RIPEMD160 The input is hashed using RIPEMD-160.	**167** in **0xa7** hash _SHA1 The input is hashed using SHA-1.											**97** Nothing **0x61** Nothing _NOP Does nothing.	**01-75** **0x01-0x4d** PUSHDATA BY Bytelength The next opcode bytes is data to be pushed onto the stack	**76** (special) **0x4c** data _PUSHDATA1 The next byte contains the number of bytes to be pushed onto the stack.	**77** (special) **0x4d** data _PUSHDATA2 The next two bytes contain the number of bytes to be pushed onto the stack.	**253** **0xfd** _PUBKEYHASH Represents a public key hashed with OP_HASH160.	**127** x n **0x7f** x1 x2 _SPLIT Splits byte sequence x at position n.
168 in **0xa8** hash _SHA256 The input is hashed using SHA-256.	**169** in **0xa9** hash _HASH160 The input is hashed twice: first with SHA-256 and then with RIPEMD-160.											**131** in **0x83** Out _INVERT Flips all of the bits in the input.	**98** Nothing **0x62** Version _VER Puts the version of the protocol under which this transaction will be evaluated onto the stack.	**78** (special) **0x4e** data PUSHDATA4 The next four bytes contain the number of bytes to be pushed onto the stack.	**79** Nothing **0x4f** -1 _1NEGATE The number -1 is pushed onto the stack.	**254** **0xfe** _PUBKEY Represents a public key compatible with OP_CHECKSIG.	**128** a b **0x80** Out _NUM2BIN Converts numeric value a into byte sequence of length b.
170 in **0xaa** hash _HASH256 The input is hashed two times with SHA-256.	**171** nothing **0xab** nothing _CODESEPARATOR All of the signature checking words will only match signatures to the data	**139** in **0x8b** Out _1ADD 1 is added to the input.	**140** in **0x8c** Out _1SUB 1 is subtracted from the input.	**141** in **0x8d** Out _2MUL The input is multiplied by 2.	**142** in **0x8e** Out _2DIV The input is divided by 2.	**143** in **0x8f** Out _NEGATIVE The sign of the input is flipped.	**144** in **0x90** Out _ABS The input is made positive.	**145** in **0x91** true/talse _NOT If the input is 0 or 1, it is flipped. Otherwise the output will be 0.	**146** in **0x92** true/talse _0NOTEQUAL Returns 0 if the input is 0. 1 otherwise.	**147** a b **0x93** Out _ADD a is added to b.	**148** a b **0x94** Out _SUB b is subtracted from a.	**132** x1 x2 **0x84** Out _AND Boolean and between each bit in the inputs.	**99** expression **0x63** IF _IF If the top stack value is TRUE, statement 1 is executed.	**100** expression **0x64** NOTIF _NOTIF If the top stack value is FALSE, statement 1 is executed.	**81** Nothing **0x51** 1 _TRUE The number 1 is pushed onto the stack.	**255** **0xff** _INVALIDOP Matches any opcode that is not yet assigned.	**129** x **0x81** Out _BIN2NUM Converts byte sequence x into a numeric value.
172 sig pubkey **0xac** true/false _CHECKSIG The entire transaction's outputs, inputs, and script (from the most recently-	**173** sig pubkey **0xad** noth,/fail _CHECKSIGVERIFY Same as OP_CHECKSIG, but OP_VERIFY is executed afterward.	**149** a b **0x95** Out _MUL a is multiplied by b.	**150** a b **0x96** Out _DIV a is divided by b.	**151** a b **0x97** Out _MOD Returns the remainder after dividing a by b.	**152** a b **0x98** Out _LSHIFT Logical left shift b bits. Sign data is discarded	**153** a b **0x99** Out _RSHIFT Logical right shift b bits. Sign data is discarded	**154** a b **0x9a** true/talse _BOOLAND If both a and b are not 0, the output is 1. Otherwise 0.	**155** a b **0x9b** true/talse _BOOLOR If a or b is not 0, the output is 1. Otherwise 0.	**156** a b **0x9c** true/talse _NUMEQUAL Returns 1 if the numbers are equal, 0 otherwise.	**157** a b **0x9d** Nothi./fail _NUMEQUALVERIFY Same as OP_NUMEQUAL, but runs OP_VERIFY afterward.	**158** a b **0x9e** true/talse _NUMNOTEQUAL Returns 1 if the numbers are not equal, 0 otherwise.	**133** x1 x2 **0x85** Out _OR Boolean or between each bit in the inputs.	**134** x1 x2 **0x86** Out _XOR Boolean exclusive or between each bit in the inputs.	**101** NA **0x64** _VERIF Do not use	**102** NA **0x65** _VERNOTIF Do not use	Constants	**130** in **0x82** in Size _SIZE Pushes the string length of the top element of the stack
174 x sig1 sig2... **0xae** true/false _CHECKMULTISIG Compares the first signature against each public key until it finds an ECDSA match.	**175** x sig1 sig2... **0xaf** noth,/fail _CHECKMULTISIGVER Same as OP_CHECKMULTISIG, but OP_VERIFY is executed afterward.	Stacks	**159** in **0x9f** true/talse _LESSTHAN Returns 1 if a is less than b, 0 otherwise.	**160** in **0xa0** true/talse _GREATERTHAN Returns 1 if a is greater than b, 0 otherwise.	**161** in **0xa1** true/talse _LESSTHANOREQ Returns 1 if a is less than or equal to b, 0 otherwise.	**162** in **0xa2** true/talse _GREATERTHANO Returns 1 if a is greater than or equal to b, 0 otherwise.	**163** in **0xa3** Out _MIN Returns the smaller of a and b.	**164** in **0xa4** Out _MAX Returns the larger of a and b.	**165** x min max **0xa5** true/talse _WITHIN Returns 1 if x is within the specified range (left-inclusive), 0 otherwise.	**177** x **0xb1** fail _CHECKLOCKTIMEVERIFY if the top stack item is greater than the transaction's nLockTime field, otherwise script	**178** x **0xb1** fail _CHECKSEQUENCEVERIFY if the relative lock time of the input is not equal to or longer than the value of the top stack item,	**135** x1 x2 **0x87** true/false _IQUAL Returns 1 if the inputs are exactly equal, 0 otherwise.	**136** x1 x2 **0x88** Noth./fail _EQUALVERIFY Same as OP_EQUAL, but runs OP_VERIFY afterward.	**103** expression **0x66** ELSE _ELSE If the preceding IF or NOTIF check was not valid then statement is executed.	**104** expression **0x67** ENDIF _ENDIF Ends an if/else block. All blocks must end, or the transaction is invalid.	**105** true/false **0x68** Noth./fail _VERIFY Marks transaction as invalid if top stack value is not true. The top stack value is removed.	**106** Nothing **0x69** EndScript _RETURN OP_RETURN can also be used to create "False Return" outputs with a scriptPubKey consisting of
210 index **0xd2** script _OUTPUTTOKENC Pop the top item from the stack as an output index (VM Number). Push the token commitment of the output at that	**211** index **0xd3** number _OUTPUTTOKENAM Pop the top item from the stack as an output index (VM Number). Push the fungible token amount of the output at that index to the stack	**125** x1x2 **0x7d** x2x1x2 _TUCK The item at the top of the stack is copied and inserted before the second-to-top item.	**80** **0x50** _RESERVED Transaction is invalid unless occuring in an unexecuted OP_IF branch.	**137** **0x89** _RESERVED1 Do not use	**138** **0x8a** _RESERVED2 Do not use	**176** **0xb0** _NOP1 Previously reserved for OP_EVAL	**179** **0xb3** _NOP4 Ignored. Does not mark transaction as invalid	**180** **0xb4** _NOP5 Ignored. Does not mark transaction as invalid	**181** **0xb5** _NOP6 Ignored. Does not mark transaction as invalid	**182** **0xb6** _NOP7 Ignored. Does not mark transaction as invalid	**183** **0xb7** _NOP8 Ignored. Does not mark transaction as invalid	**184** **0xb8** _NOP9 Ignored. Does not mark transaction as invalid	**185** **0xb9** _NOP10 Ignored. Does not mark transaction as invalid	**186** sig msg **0xae** true/false _CHECKDATASIG Check if signature is valid for message and a public key,	**187** sig msg **0xae** noth,/fail _CHECKDATASIGVE Same as OP_CHECKDATASIG, but runs OP_VERIFY afterward.	**188** x **0x82** Out _REVERSEBYTES Reverses the order of the bytes in byte sequence x so that the first byte is now its last byte, th	
192 nothing **0xc0** number _INPUTINDEX Push the index of the input being evaluated to the stack as a Script Number.	**193** nothing **0xc1** script _ACTIVEBYTECODE Push the bytecode currently being evaluated, beginning after the last executed OP_CODESEPARATO	**194** nothing **0xc2** number _TXVERSION Push the version of the current transaction to the stack as a Script Number.	**195** nothing **0xc3** number _TXINPUTCOUNT Push the count of inputs in the current transaction to the stack as a Script Number.	**196** nothing **0xc4** number _TXOUTPUTCOUN Push the count of outputs in the current transaction to the stack as a Script Number.	**197** nothing **0xc5** number _TXLOCKTIME Push the locktime of the current transaction to the stack as a Script Number.	**198** index **0xc6** number _UTXOVALUE Push the value (in satoshis) of the Unspent Transaction Output (UTXO) spent by that input to the	**199** index **0xc7** script _UTXOBYTECODE Push the full locking bytecode of the Unspent Transaction Output (UTXO) spent by that input to th	**200** index **0xc8** hash _OUTPOINTTXHAS From that input, push the outpoint transaction hash – the hash of the transaction which	**201** index **0xc9** number _OUTPOINTINDEX From that input, push the outpoint index – the index of the output in the transaction which	**202** index **0xca** script _INPUTBYTECODE Push the unlocking bytecode of the input at that index to the stack.	**203** index **0xcb** number _INPUTSEQUENCE Push the sequence number of the input at that index to the stack as a Script Number.	**204** index **0xcc** number _OUTPUTVALUE Push the value (in satoshis) of the output at that index to the stack as a Script Number.	**205** index **0xcd** script _OUTPUTBYTECO Push the locking bytecode of the output at that index to the stack.	**206** index **0xce** script _UTXOTOKENCATEG Pop the top item from the stack as an input index (VM Number). If the Unspent Transaction Output (UTXO) spent by	**207** index **0xcf** script _UTXOTOKENCOMM Pop the top item from the stack as an input index (VM Number). Push the token commitment of the	**208** index **0xd0** number _UTXOTOKENAMOU Pop the top item from the stack as an input index (VM Number). Push the fungible token amount of th	**209** index **0xd1** script _OUTPUTTOKENC Pop the top item from the stack as an output index (VM Number). If the output at that index includes
		Constants	**82** Nothing **0x52** 2 _2 The number in the word name (2) is pushed onto the stack.	**83** Nothing **0x53** 3 _3 The number in the word name (3) is pushed onto the stack.	**84** Nothing **0x54** 4 _4 The number in the word name (4) is pushed onto the stack.	**85** Nothing **0x55** 5 _5 The number in the word name (5) is pushed onto the stack.	**86** Nothing **0x56** 6 _6 The number in the word name (6) is pushed onto the stack.	**87** Nothing **0x57** 7 _7 The number in the word name (7) is pushed onto the stack.	**88** Nothing **0x58** 8 _8 The number in the word name (8) is pushed onto the stack.	**89** Nothing **0x59** 9 _9 The number in the word name (9) is pushed onto the stack.	**90** Nothing **0x5A** 10 _10 The number in the word name (10) is pushed onto the stack.	**91** Nothing **0x5B** 11 _11 The number in the word name (11) is pushed onto the stack.	**9C** Nothing **0x5C** 12 _12 The number in the word name (12) is pushed onto the stack.	**93** Nothing **0x5D** 13 _13 The number in the word name (13) is pushed onto the stack.	**94** Nothing **0x5E** 14 _14 The number in the word name (14) is pushed onto the stack.	**95** Nothing **0x5F** 15 _15 The number in the word name (15) is pushed onto the stack.	**96** Nothing **0x60** 16 _16 The number in the word name (16) is pushed onto the stack.
Stack **107** x1 **0x6b** (alt)x1 _TOALTSTACK Puts the input onto the top of the alt stack. Removes it from the main stack.	**108** (alt)x1 **0x6c** x1 _FROMALTSTACK Puts the input onto the top of the main stack. Removes it from the alt stack.	**109** x1 x2 **0x6d** Nothing _2DROP Removes the top two stack items.	**110** x1x2 **0x6e** x1x2x1x2 _2DUP Duplicates the top two stack items.	**111** x1x2x3 **0x6f** x1x2x3x1x _3DUP Duplicates the top three stack items.	**112** x1x2x3x4 **0x70** x1x2x3x4x _2OVER Copies the pair of items two spaces back in the stack to the front.	**113** x1x2x3x4x5x6 **0x71** x3x4x5x6x _2ROT The fifth and sixth items back are moved to the top of the stack.	**114** x1x2x3x4 **0x72** x3x4x1x2 _2SWAP Swaps the top two pairs of items.	**115** x **0x73** x/xx _IFDUP If the top stack value is not 0, duplicate it.	**116** Nothing **0x74** stack size _DEPTH Counts the number of stack items onto the stack and places the value on the top	**117** x **0x75** Nothing _DROP Removes the top stack item.	**118** x **0x76** xx _DUP Duplicates the top stack item.	**119** x1x2 **0x77** x2 _NIP Removes the second-to-top stack item.	**120** x1x2 **0x78** x1x2x1 _OVER Copies the second-to-top stack item to the top.	**121** xn...x2x1x0 **0x79** xn...x2x1x0xn _PICK The item n back in the stack is copied to the top.	**122** xn...x2x1x0 **0x7a** ..x2x1x0xn _ROLL The item n back in the stack is moved to the top.	**123** x1x2x3 **0x7b** x2x3x1 _ROT The top three items on the stack are rotated to the left.	**124** x1x2 **0x7c** x2x1 _SWAP The top two items on the stack are swapped.