CORE CODES TABLE
BTC

Copyright © BSV Colombia, pelaezj1, 2024

Category	Category
Cryptography	Data Manipulation
Arithmetic	Pseudo-Words
Stack	Constants
Bitwise Logic	Flow Control
Reserved Words	Dangerous

Operational Code

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

https://en.bitcoin.it/wiki/Script

Row 1

- **0** (Nothing / empty) **0x00** _FALSE — An empty array of bytes is pushed onto the stack.
- **126** (x1 x2 / Out) **0x7e** _CAT — Concatenates two strings.

Row 2

- **166** (in / hash) **0xa6** _RIPEMD160 — The input is hashed using RIPEMD-160.
- **167** (in / hash) **0xa7** _SHA1 — The input is hashed using SHA-1.
- **97** (Nothing / Nothing) **0x61** _NOP — Does nothing.
- **01-75** **0x01-0x4d** PUSHDATA BY Bytelength — The next opcode bytes is data to be pushed onto the stack
- **76** ((special) / data) **0x4c** _PUSHDATA1 — The next byte contains the number of bytes to be pushed onto the stack.
- **77** ((special) / data) **0x4d** _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 / x1 x2) **0x7f** _SPLIT — Splits byte sequence x at position n.

Row 3

- **168** (in / hash) **0xa8** _SHA256 — The input is hashed using SHA-256.
- **169** (in / hash) **0xa9** _HASH160 — The input is hashed twice: first with SHA-256 and then with RIPEMD-160.
- **131** (in / Out) **0x83** _INVERT — Flips all of the bits in the input.
- **98** (Nothing / Version) **0x62** _VER — Puts the version of the protocol under which this transaction will be evaluated onto the stack.
- **78** ((special) / data) **0x4e** PUSHDATA4 — The next four bytes contain the number of bytes to be pushed onto the stack.
- **79** (Nothing / -1) **0x4f** _1NEGATE — The number -1 is pushed onto the stack.
- **254** **0xfe** _PUBKEY — Represents a public key compatible with OP_CHECKSIG.
- **128** (in size / Out) **0x80** _LEFT — Converts numeric value a into byte sequence of length b.

Row 4

- **170** (in / hash) **0xaa** _HASH256 — The input is hashed two times with SHA-256.
- **171** (nothing / nothing) **0xab** _CODESEPARATOR — All of the signature checking words will only match signatures to the
- **139** (in / Out) **0x8b** _1ADD — 1 is added to the input.
- **140** (in / Out) **0x8c** _1SUB — 1 is subtracted from the input.
- **141** (in / Out) **0x8d** _2MUL — The input is multiplied by 2.
- **142** (in / Out) **0x8e** _2DIV — The input is divided by 2.
- **143** (in / Out) **0x8f** _NEGATIVE — The sign of the input is flipped.
- **144** (in / Out) **0x90** _ABS — The input is made positive.
- **145** (in / Out) **0x91** _NOT — If the input is 0 or 1, it is flipped. Otherwise the output will be 0.
- **146** (in / Out) **0x92** _0NOTEQUAL — Returns 0 if the input is 0. 1 otherwise.
- **147** (a b / Out) **0x93** _ADD — a is added to b.
- **148** (a b / Out) **0x94** _SUB — b is subtracted from a.
- **132** (x1 x2 / Out) **0x84** _AND — Boolean and between each bit in the inputs.
- **99** (expression / IF) **0x63** _IF — If the top stack value is TRUE, statement 1 is executed.
- **100** (expression / NOTIF) **0x64** _NOTIF — If the top stack value is FALSE, statement 1 is executed.
- **81** (Nothing / 1) **0x51** _TRUE — The number 1 is pushed onto the stack.
- **255** **0xff** _INVALIDOP — Matches any opcode that is not yet assigned.
- **129** (in size / Out) **0x81** _RIGHT — Converts byte sequence x into a numeric value.

Row 5

- **172** (sig pubkey / true/false) **0xac** _CHECKSIG — The entire transaction's outputs, inputs, and script (from the most recently-
- **173** (sig pubkey / noth,/fail) **0xad** _CHECKSIGVERIFY — Same as OP_CHECKSIG, but OP_VERIFY is executed afterward.
- **149** (a b / Out) **0x95** _MUL — a is multiplied by b.
- **150** (a b / Out) **0x96** _DIV — a is divided by b.
- **151** (a b / Out) **0x97** _MOD — Returns the remainder after dividing a by b.
- **152** (a b / Out) **0x98** _LSHIFT — Logical left shift b bits. Sign data is discarded
- **153** (a b / Out) **0x99** _RSHIFT — Logical right shift b bits. Sign data is discarded
- **154** (a b / Out) **0x9a** _BOOLAND — If both a and b are not 0, the output is 1. Otherwise 0.
- **155** (a b / Out) **0x9b** _BOOLOR — If a or b is not 0, the output is 1. Otherwise 0.
- **156** (a b / Nothi./fail) **0x9c** _NUMEQUAL — Returns 1 if the numbers are equal, 0 otherwise.
- **157** (a b / Out) **0x9d** _NUMEQUALVERIFY — Same as OP_NUMEQUAL, but runs OP_VERIFY afterward.
- **158** (a b / Out) **0x9e** _NUMNOTEQUAL — Returns 1 if the numbers are not equal, 0 otherwise.
- **133** (x1 x2 / Out) **0x85** _OR — Boolean or between each bit in the inputs.
- **134** (x1 x2 / Out) **0x86** _XOR — Boolean exclusive or between each bit in the inputs.
- **101** (NA) **0x64** _VERIF — Disabled
- **102** (NA) **0x65** _VERNOTIF — Disabled
- **Constants**
- **130** (in / in Size) **0x82** _SIZE — Pushes the string length of the top element of the stack

Row 6

- **174** (x sig1 sig2... / true/false) **0xae** _CHECKMULTISIG — Compares the first signature against each public key until it finds an ECDSA match.
- **175** (x sig1 sig2... / noth,/fail) **0xaf** _CHECKMULTISIGVERIFY — Same as OP_CHECKMULTISIG, but OP_VERIFY is executed afterward.
- **Stacks**
- **159** (in / Out) **0x9f** _LESSTHAN — Returns 1 if a is less than b, 0 otherwise.
- **160** (in / Out) **0xa0** _GREATERTHAN — Returns 1 if a is greater than b, 0 otherwise.
- **161** (in / Out) **0xa1** _LESSTHANOREQ — Returns 1 if a is less than or equal to b, 0 otherwise.
- **162** (in / Out) **0xa2** _GREATERTHANO — Returns 1 if a is greater than or equal to b, 0 otherwise.
- **163** (in / Out) **0xa3** _MIN — Returns the smaller of a and b.
- **164** (in / Out) **0xa4** _MAX — Returns the larger of a and b.
- **165** (x min max / Out) **0xa5** _WITHIN — Returns 1 if x is within the specified range (left-inclusive), 0 otherwise.
- **177** (x / fail) **0xb1** _CHECKLOCKTIMEVERIFY — if the top stack item is greater than the transaction's nLockTime field, otherwise script evaluation continues as
- **178** (x / fail) **0xb1** _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 / true/false) **0x87** _IQUAL — Returns 1 if the inputs are exactly equal, 0 otherwise.
- **136** (x1 x2 / Noth./fail) **0x88** _EQUALVERIFY — Same as OP_EQUAL, but runs OP_VERIFY afterward.
- **103** (expression / ELSE) **0x66** _ELSE — If the preceding IF or NOTIF check was not valid then statement is executed.
- **104** (expression / ENDIF) **0x67** _ENDIF — Ends an if/else block. All blocks must end, or the transaction is invalid.
- **105** (true/false / Noth./fail) **0x68** _VERIFY — Marks transaction as invalid if top stack value is not true. The top stack value is removed.
- **106** (Nothing / EndScript) **0x69** _RETURN — OP_RETURN can also be used to create "False Return" outputs with a scriptPubKey consisting of

Row 7

- **125** (x1x2 / x2x1x2) **0x7d** _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 — Transaction is invalid unless occuring in an unexecuted OP_IF branch.
- **138** **0x8a** _RESERVED2 — Transaction is invalid unless occuring in an unexecuted OP_IF branch.
- **176** **0xb0** _NOP1 — The word is ignored. Does not mark transaction as invalid.
- **179** **0xb3** _NOP4 — The word is ignored. Does not mark transaction as invalid.
- **180** **0xb4** _NOP5 — The word is ignored. Does not mark transaction as invalid.
- **181** **0xb5** _NOP6 — The word is ignored. Does not mark transaction as invalid.
- **182** **0xb6** _NOP7 — The word is ignored. Does not mark transaction as invalid.
- **183** **0xb7** _NOP8 — The word is ignored. Does not mark transaction as invalid.
- **184** **0xb8** _NOP9 — The word is ignored. Does not mark transaction as invalid.
- **185** **0xb9** _NOP10 — The word is ignored. Does not mark transaction as invalid.

Constants

- **82** (Nothing / 2) **0x52** _2 — The number in the word name (2) is pushed onto the stack.
- **83** (Nothing / 3) **0x53** _3 — The number in the word name (3) is pushed onto the stack.
- **84** (Nothing / 4) **0x54** _4 — The number in the word name (4) is pushed onto the stack.
- **85** (Nothing / 5) **0x55** _5 — The number in the word name (5) is pushed onto the stack.
- **86** (Nothing / 6) **0x56** _6 — The number in the word name (6) is pushed onto the stack.
- **87** (Nothing / 7) **0x57** _7 — The number in the word name (7) is pushed onto the stack.
- **88** (Nothing / 8) **0x58** _8 — The number in the word name (8) is pushed onto the stack.
- **89** (Nothing / 9) **0x59** _9 — The number in the word name (9) is pushed onto the stack.
- **90** (Nothing / 10) **0x5A** _10 — The number in the word name (10) is pushed onto the stack.
- **91** (Nothing / 11) **0x5B** _11 — The number in the word name (11) is pushed onto the stack.
- **9C** (Nothing / 12) **0x5C** _12 — The number in the word name (12) is pushed onto the stack.
- **93** (Nothing / 13) **0x5D** _13 — The number in the word name (13) is pushed onto the stack.
- **94** (Nothing / 14) **0x5E** _14 — The number in the word name (14) is pushed onto the stack.
- **95** (Nothing / 15) **0x5F** _15 — The number in the word name (15) is pushed onto the stack.
- **96** (Nothing / 16) **0x60** _16 — The number in the word name (16) is pushed onto the stack.

Stack

- **107** (x1 / (alt)x1) **0x6b** _TOALTSTACK — Puts the input onto the top of the alt stack. Removes it from the main stack.
- **108** ((alt)x1 / x1) **0x6c** _FROMALTSTACK — Puts the input onto the top of the main stack. Removes it from the alt stack.
- **109** (x1 x2 / Nothing) **0x6d** _2DROP — Removes the top two stack items.
- **110** (x1x2 / x1x2x1x2) **0x6e** _2DUP — Duplicates the top two stack items.
- **111** (x1x2x3 / x1x2x3x1x) **0x6f** _3DUP — Duplicates the top three stack items.
- **112** (x1x2x3x4 / x1x2x3x4x) **0x70** _2OVER — Copies the pair of items two spaces back in the stack to the front.
- **113** (x1x2x3x4x5x6 / x3x4x5x6x) **0x71** _2ROT — The fifth and sixth items back are moved to the top of the stack.
- **114** (x1x2x3x4 / x3x4x1x2) **0x72** _2SWAP — Swaps the top two pairs of items.
- **115** (x / x/xx) **0x73** _IFDUP — If the top stack value is not 0, duplicate it.
- **116** (Nothing / stack size) **0x74** _DEPTH — Counts the number of stack items onto the stack and places the value on the top
- **117** (x / Nothing) **0x75** _DROP — Removes the top stack item.
- **118** (x / xx) **0x76** _DUP — Duplicates the top stack item.
- **119** (x1x2 / x2) **0x77** _NIP — Removes the second-to-top stack item.
- **120** (x1x2 / x1x2x1) **0x78** _OVER — Copies the second-to-top stack item to the top.
- **121** (xn...x2x1x0 / xn...x2x1x) **0x79** _PICK — The item n back in the stack is copied to the top.
- **122** (xn...x2x1x0 / ..x2x1x0xn) **0x7a** _ROLL — The item n back in the stack is moved to the top.
- **123** (x1x2x3 / x2x3x1) **0x7b** _ROT — The top three items on the stack are rotated to the left.
- **124** (x1x2 / x2x1) **0x7c** _SWAP — The top two items on the stack are swapped.