Word	Input	Output	Description
OP_0, OP_FALSE	Nothing.	(empty value)	An empty array of bytes is pushed onto the stack. (This is not a no-op: an item is added to the stack.)
OP_PUSHDATA1	(special)	data	The next byte contains the number of bytes to be pushed onto the stack.
OP_PUSHDATA2	(special)	data	The next two bytes contain the number of bytes to be pushed onto the stack in little endian order.
OP_PUSHDATA4	(special)	data	The next four bytes contain the number of bytes to be pushed onto the stack in little endian order.
OP_1NEGATE	Nothing.	-1	The number -1 is pushed onto the stack.
OP_1, OP_TRUE	Nothing.	1	The number 1 is pushed onto the stack.
OP_2-OP_16	Nothing.	2-16	The number in the word name (2-16) is pushed onto the stack.
OP_NOP	Nothing	Nothing	Does nothing.
OP_IF	<expression> if [statements] [else [statements]]* endif	If the top stack value is not False, the statements are executed. The top stack value is removed.	
OP_NOTIF	<expression> notif [statements] [else [statements]]* endif	If the top stack value is False, the statements are executed. The top stack value is removed.	
OP_ELSE	<expression> if [statements] [else [statements]]* endif	If the preceding OP_IF or OP_NOTIF or OP_ELSE was not executed then these statements are and if the preceding OP_IF or OP_NOTIF or OP_ELSE was executed then these statements are not.	
OP_ENDIF	<expression> if [statements] [else [statements]]* endif	Ends an if/else block. All blocks must end, or the transaction is **invalid**. An OP_ENDIF without OP_IF earlier is also **invalid**.	
OP_VERIFY	True / false	Nothing / *fail*	**Marks transaction as invalid** if top stack value is not true. The top stack value is removed.
OP_RETURN	Nothing	*fail*	**Marks transaction as invalid**. A standard way of attaching extra data to transactions is to add a zero-value output with a scriptPubKey consisting of OP_RETURN followed by exactly one pushdata op. Such outputs are provably unspendable, reducing their cost to the network. Currently it is usually considered non-standard (though valid) for a transaction to have more than one OP_RETURN output or an OP_RETURN output with more than one pushdata op.
OP_IFDUP	x	x / x x	If the top stack value is not 0, duplicate it.
OP_DEPTH	Nothing	<Stack size>	Puts the number of stack items onto the stack.
OP_DROP	x	Nothing	Removes the top stack item.
OP_DUP	x	x x	Duplicates the top stack item.
OP_OVER	x1 x2	x1 x2 x1	Copies the second-to-top stack item to the top.
OP_PICK	xn ... x2 x1 x0 <n>	xn ... x2 x1 x0 xn	The item *n* back in the stack is copied to the top.
OP_ROT	x1 x2 x3	x2 x3 x1	The top three items on the stack are rotated to the left.
OP_SWAP	x1 x2	x2 x1	The top two items on the stack are swapped.
OP_2DROP	x1 x2	Nothing	Removes the top two stack items.
OP_2DUP	x1 x2	x1 x2 x1 x2	Duplicates the top two stack items.
OP_2SWAP	x1 x2 x3 x4	x3 x4 x1 x2	Swaps the top two pairs of items.
OP_EQUAL	x1 x2	True / false	Returns 1 if the inputs are exactly equal, 0 otherwise.
OP_EQUALVERIFY	x1 x2	Nothing / *fail*	Same as OP_EQUAL, but runs OP_VERIFY afterward.
OP_NEGATE	in	out	The sign of the input is flipped.
OP_ABS	in	out	The input is made positive.
OP_NOT	in	out	If the input is 0 or 1, it is flipped. Otherwise the output will be 0.
OP_ADD	a b	out	a is added to b.
OP_SUB	a b	out	b is subtracted from a.
OP_NUMEQUAL	a b	out	Returns 1 if the numbers are equal, 0 otherwise.

OP_NUMEQUALVERIFY	a b	Nothing / *fail*	Same as OP_NUMEQUAL, but runs OP_VERIFY afterward.
OP_NUMNOTEQUAL	a b	out	Returns 1 if the numbers are not equal, 0 otherwise.
OP_LESSTHAN	a b	out	Returns 1 if a is less than b, 0 otherwise.
OP_GREATERTHAN	a b	out	Returns 1 if a is greater than b, 0 otherwise.
OP_LESSTHANOREQUAL	a b	out	Returns 1 if a is less than or equal to b, 0 otherwise.
OP_GREATERTHANOREQUAL	a b	out	Returns 1 if a is greater than or equal to b, 0 otherwise.
OP_MIN	a b	out	Returns the smaller of a and b.
OP_MAX	a b	out	Returns the larger of a and b.
OP_RIPEMD160	in	hash	The input is hashed using RIPEMD-160.
OP_SHA1	in	hash	The input is hashed using SHA-1.
OP_SHA256	in	hash	The input is hashed using SHA-256.
OP_HASH160	in	hash	The input is hashed twice: first with SHA-256 and then with RIPEMD-160.
OP_HASH256	in	hash	The input is hashed two times with SHA-256.
OP_CHECKLOCKTIMEVERIFY	x	x / fail	Marks transaction as invalid if the top stack item is greater than the transaction's **nLockTime** field, otherwise script evaluation continues as though an OP_NOP was executed. Transaction is also invalid if 1. the stack is empty; or 2. the top stack item is negative; or 3. the top stack item is greater than or equal to 500000000 while the transaction's nLockTime field is less than 500000000, or vice versa; or 4. the input's nSequence field is equal to 0xffffffff.
OP_CHECKSIG	sig pubkey	True / false	The entire transaction's outputs, inputs, and script (from the most recently-executed OP_CODESEPARATOR to the end) are hashed. The signature used by OP_CHECKSIG must be a valid signature for this hash and public key. If it is, 1 is returned, 0 otherwise.
OP_CHECKSIGVERIFY	sig pubkey	Nothing / *fail*	Same as OP_CHECKSIG, but OP_VERIFY is executed afterward.
OP_CHECKMULTISIG	x sig1 sig2 ... <number of signatures> pub1 pub2 <number of public keys>	True / False	Compares the first signature against each public key until it finds an ECDSA match. Starting with the subsequent public key, it compares the second signature against each remaining public key until it finds an ECDSA match. The process is repeated until all signatures have been checked or not enough public keys remain to produce a successful result. All signatures need to match a public key. Because public keys are not checked again if they fail any signature comparison, signatures must be placed in the scriptSig using the same order as their corresponding public keys were placed in the scriptPubKey or redeemScript. If all signatures are valid, 1 is returned, 0 otherwise. Due to a bug, one extra unused value is removed from the stack.
OP_CHECKMULTISIGVERIFY	x sig1 sig2 ... <number of signatures> pub1 pub2 ... <number of public keys>	Nothing / *fail*	Same as OP_CHECKMULTISIG, but OP_VERIFY is executed afterward.

nLockTime field represents block number or timestamp at which the transacton or UTXO is unlocked (ready to be spent). The table specifies how time is determined:

Value	Description
0	Transaction/UTXO not bounded by time.
< 500,000,000	Block number at which the transaction/UTXO is unlocked.
>= 500,000,000	UNIX timestamp at which the transaction/UTXO in unlocked.