EVM Opcode Reference for Osaka Fork

Sources: evm.codes

Stack shown top-first. Stack In = values popped (top → bottom). Stack Out = values pushed (top → bottom).

DUP/SWAP: DUPn pops n, pushes n+1 · SWAPn pops n+1, pushes n+1 (net zero, reorders)

Arithmetic (0x00–0x0B)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x00	STOP	0	—	0	—	Halt execution
0x01	ADD	2	a, b	1	a + b	Integer addition mod 2^{256} (silently wraps)
0x02	MUL	2	a, b	1	a * b	Integer multiplication mod 2^{256} (silently wraps)
0x03	SUB	2	a, b	1	a - b	Integer subtraction mod 2^{256} (silently wraps)
0x04	DIV	2	a, b	1	a / b (0 if b=0)	Unsigned integer division
0x05	SDIV	2	a, b	1	a / b signed	Signed integer division (two’s complement)
0x06	MOD	2	a, b	1	a % b (0 if b=0)	Unsigned integer modulo
0x07	SMOD	2	a, b	1	a % b signed	Signed integer modulo (two’s complement)
0x08	ADDMOD	3	a, b, N	1	(a + b) % N (0 if N=0)	Addition modulo N
0x09	MULMOD	3	a, b, N	1	(a * b) % N (0 if N=0)	Multiplication modulo N
0x0A	EXP	2	a, exponent	1	a ** exponent	Exponentiation mod 2^{256} (silently wraps)
0x0B	SIGNEXTEND	2	b, x	1	signext(x, (b+1)*8)	Sign-extend x from (b+1)×8 bits to 256 bits

Comparison & Bitwise Logic (0x10–0x1E)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x10	LT	2	a, b	1	a < b ? 1 : 0	Unsigned less-than
0x11	GT	2	a, b	1	a > b ? 1 : 0	Unsigned greater-than
0x12	SLT	2	a, b	1	a < b ? 1 : 0	Signed less-than (two’s complement)
0x13	SGT	2	a, b	1	a > b ? 1 : 0	Signed greater-than (two’s complement)
0x14	EQ	2	a, b	1	a == b ? 1 : 0	Equality check

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x15	ISZERO	1	a	1	a == 0 ? 1 : 0	Push 1 if input is zero, 0 otherwise
0x16	AND	2	a, b	1	a & b	Bitwise AND
0x17	OR	2	a, b	1	a | b	Bitwise OR
0x18	XOR	2	a, b	1	a ^ b	Bitwise XOR
0x19	NOT	1	a	1	~a	Bitwise NOT
0x1A	BYTE	2	i, x	1	(x » (248-i*8)) & 0xFF	Extract byte i (0=MSB) from 32-byte word
0x1B	SHL	2	shift, a	1	a « shift	Left shift (EIP-145)
0x1C	SHR	2	shift, a	1	a » shift	Logical (unsigned) right shift (EIP-145)
0x1D	SAR	2	shift, a	1	a »> shift	Arithmetic (signed) right shift (EIP-145)
0x1E	CLZ	1	x	1	clz(x)	Count leading zero bits in x; returns 256 if x = 0 (EIP-7939, Osaka)

Hashing (0x20)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x20	KECCAK256	2	offset, size	1	keccak256(mem[offset..offset+size])	Keccak-256 hash of memory region

Environment Information (0x30–0x3F)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x30	ADDRESS	0	—	1	address(this)	Address of the currently executing account
0x31	BALANCE	1	addr	1	addr.balance	Wei balance of addr
0x32	ORIGIN	0	—	1	tx.origin	Transaction originator address
0x33	CALLER	0	—	1	msg.sender	Direct caller address
0x34	CALLVALUE	0	—	1	msg.value	Value sent with this call in wei
0x35	CALLDATALOAD	1	i	1	msg.data[i..i+32]	Read 32 bytes of calldata at byte offset
0x36	CALLDATASIZE	0	—	1	len(msg.data)	Size of calldata in bytes
0x37	CALLDATACOPY	3	dstOffset, offset, size	0	mem[dstOffset..] = calldata	Copy calldata into memory
0x38	CODESIZE	0	—	1	len(this.code)	Size of executing contract’s code in bytes
0x39	CODECOPY	3	dstOffset, offset, size	0	mem[dstOffset..] = code	Copy executing contract’s code into memory
0x3A	GASPRICE	0	—	1	tx.gasprice	Effective gas price in wei per gas
0x3B	EXTCODESIZE	1	addr	1	len(addr.code)	Code size of external account in bytes
0x3C	EXTCODECOPY	4	addr, dstOffset, offset, size	0	mem[dstOffset..] = addr.code	Copy external account’s code into memory
0x3D	RETURNDATASIZE	0	—	1	len(returndata)	Size of return data from most recent sub-call
0x3E	RETURNDATACOPY	3	dstOffset, offset, size	0	mem[dstOffset..] = returndata	Copy return data buffer into memory
0x3F	EXTCODEHASH	1	addr	1	keccak256(addr.code)	Keccak-256 hash of external account’s code (EIP-1052)

Block Information (0x40–0x4A)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x40	BLOCKHASH	1	blockNum	1	blockhash(blockNum)	Hash of one of the 256 most recent complete blocks; 0 if out of range
0x41	COINBASE	0	—	1	block.coinbase	Block beneficiary (validator) address
0x42	TIMESTAMP	0	—	1	block.timestamp	Block timestamp in Unix seconds
0x43	NUMBER	0	—	1	block.number	Current block number
0x44	PREVRANDAO	0	—	1	block.prevrandao	Previous block's RANDAO mix (was DIFFICULTY pre-Merge; EIP-4399)
0x45	GASLIMIT	0	—	1	block.gaslimit	Current block's gas limit
0x46	CHAINID	0	—	1	chain_id	Chain ID (EIP-1344)
0x47	SELFBALANCE	0	—	1	address(this).balance	Balance of the executing contract in wei (EIP-1884)
0x48	BASEFEE	0	—	1	block.basefee	Base fee per gas of the current block in wei (EIP-3198)
0x49	BLOBHASH	1	i	1	tx.blob_hashes[i]	Versioned hash of blob at index i in the transaction; 0 if out of range
0x4A	BLOBBASEFEE	0	—	1	block.blobbasefee	Current blob base fee in wei

Stack, Memory, Storage & Flow (0x50–0x5F)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x50	POP	1	a	0	—	Remove and discard top stack item
0x51	MLOAD	1	offset	1	mem[offset..offset+32]	Load 32-byte word from memory
0x52	MSTORE	2	offset, value	0	mem[offset..offset+32]=value	Store 32-byte word to memory
0x53	MSTORE8	2	offset, value	0	mem[offset] = value & 0xFF	Store lowest byte to memory
0x54	SLOAD	1	key	1	storage[key]	Load word from persistent storage
0x55	SSTORE	2	key, value	0	storage[key] = value	Store word to persistent storage
0x56	JUMP	1	dest	0	PC = dest	Unconditional jump (dest must be JUMPDEST)
0x57	JUMPI	2	dest, cond	0	PC = cond≠0 ? dest : PC+1	Conditional jump: if cond≠0, set PC to dest (must be a JUMPDEST); otherwise fall through to next instruction
0x58	PC	0	—	1	PC	Program counter prior to this instruction
0x59	MSIZE	0	—	1	len(mem)	Size of active memory in bytes (multiple of 32)
0x5A	GAS	0	—	1	gas_remaining	Gas remaining after this instruction
0x5B	JUMPDEST	0	—	0	—	Mark valid jump destination; no-op
0x5C	TLOAD	1	key	1	tstorage[key]	Load word from transient storage
0x5D	TSTORE	2	key, value	0	tstorage[key] = value	Store word to transient storage; wiped at end of transaction
0x5E	MCOPY	3	dst, src, len	0	mem[dst..]=mem[src..src+len]	Copy memory region; handles overlapping src/dst correctly
0x5F	PUSH0	0	—	1	0	Push constant 0 (EIP-3855, Shanghai)

Push Operations (0x60–0x7F)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x60	PUSH1	0	—	1	imm_{1}	Push 1-byte immediate value
0x61	PUSH2	0	—	1	imm_{2}	Push 2-byte immediate value
0x62	PUSH3	0	—	1	imm_{3}	Push 3-byte immediate value
0x63	PUSH4	0	—	1	imm_{4}	Push 4-byte immediate value
0x64	PUSH5	0	—	1	imm_{5}	Push 5-byte immediate value
0x65	PUSH6	0	—	1	imm_{6}	Push 6-byte immediate value
0x66	PUSH7	0	—	1	imm_{7}	Push 7-byte immediate value
0x67	PUSH8	0	—	1	imm_{8}	Push 8-byte immediate value
0x68	PUSH9	0	—	1	imm_{9}	Push 9-byte immediate value
0x69	PUSH10	0	—	1	imm_{10}	Push 10-byte immediate value
0x6A	PUSH11	0	—	1	imm_{11}	Push 11-byte immediate value
0x6B	PUSH12	0	—	1	imm_{12}	Push 12-byte immediate value
0x6C	PUSH13	0	—	1	imm_{13}	Push 13-byte immediate value
0x6D	PUSH14	0	—	1	imm_{14}	Push 14-byte immediate value
0x6E	PUSH15	0	—	1	imm_{15}	Push 15-byte immediate value
0x6F	PUSH16	0	—	1	imm_{16}	Push 16-byte immediate value
0x70	PUSH17	0	—	1	imm_{17}	Push 17-byte immediate value
0x71	PUSH18	0	—	1	imm_{18}	Push 18-byte immediate value
0x72	PUSH19	0	—	1	imm_{19}	Push 19-byte immediate value
0x73	PUSH20	0	—	1	imm_{20}	Push 20-byte immediate value
0x74	PUSH21	0	—	1	imm_{21}	Push 21-byte immediate value
0x75	PUSH22	0	—	1	imm_{22}	Push 22-byte immediate value
0x76	PUSH23	0	—	1	imm_{23}	Push 23-byte immediate value
0x77	PUSH24	0	—	1	imm_{24}	Push 24-byte immediate value
0x78	PUSH25	0	—	1	imm_{25}	Push 25-byte immediate value
0x79	PUSH26	0	—	1	imm_{26}	Push 26-byte immediate value
0x7A	PUSH27	0	—	1	imm_{27}	Push 27-byte immediate value
0x7B	PUSH28	0	—	1	imm_{28}	Push 28-byte immediate value
0x7C	PUSH29	0	—	1	imm_{29}	Push 29-byte immediate value
0x7D	PUSH30	0	—	1	imm_{30}	Push 30-byte immediate value
0x7E	PUSH31	0	—	1	imm_{31}	Push 31-byte immediate value
0x7F	PUSH32	0	—	1	imm_{32}	Push full 32-byte (256-bit) word

Duplication Operations (0x80–0x8F)

Stack In (top-first): a_1, a_2, …, an — Stack Out: an, a_1, a_2, …, an (copy of an pushed to top)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x80	DUP1	1	a_1	2	a_1, a_1	Duplicate 1st stack item

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x81	DUP2	2	a_1, a_2	3	a_2, a_1, a_2	Duplicate 2nd stack item
0x82	DUP3	3	a_1, a_2, a_3	4	a_3, a_1, a_2, a_3	Duplicate 3rd stack item
0x83	DUP4	4	$a_1..a_4$	5	a_4, $a_1..a_4$	Duplicate 4th stack item
0x84	DUP5	5	$a_1..a_5$	6	a_5, $a_1..a_5$	Duplicate 5th stack item
0x85	DUP6	6	$a_1..a_6$	7	a_6, $a_1..a_6$	Duplicate 6th stack item
0x86	DUP7	7	$a_1..a_7$	8	a_7, $a_1..a_7$	Duplicate 7th stack item
0x87	DUP8	8	$a_1..a_8$	9	a_8, $a_1..a_8$	Duplicate 8th stack item
0x88	DUP9	9	$a_1..a_9$	10	a_9, $a_1..a_9$	Duplicate 9th stack item
0x89	DUP10	10	$a_1..a_{10}$	11	a_{10}, $a_1..a_{10}$	Duplicate 10th stack item
0x8A	DUP11	11	$a_1..a_{11}$	12	a_{11}, $a_1..a_{11}$	Duplicate 11th stack item
0x8B	DUP12	12	$a_1..a_{12}$	13	a_{12}, $a_1..a_{12}$	Duplicate 12th stack item
0x8C	DUP13	13	$a_1..a_{13}$	14	a_{13}, $a_1..a_{13}$	Duplicate 13th stack item
0x8D	DUP14	14	$a_1..a_{14}$	15	a_{14}, $a_1..a_{14}$	Duplicate 14th stack item
0x8E	DUP15	15	$a_1..a_{15}$	16	a_{15}, $a_1..a_{15}$	Duplicate 15th stack item
0x8F	DUP16	16	$a_1..a_{16}$	17	a_{16}, $a_1..a_{16}$	Duplicate 16th stack item

Exchange Operations (0x90–0x9F)

Stack In (top-first): a_1, a_2, …, an+1 — Stack Out: an+1, a_2, …, an, a_1 (top swapped with n+1th item)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0x90	SWAP1	2	a, b	2	b, a	Exchange 1st and 2nd stack items
0x91	SWAP2	3	a, b, c	3	c, b, a	Exchange 1st and 3rd stack items
0x92	SWAP3	4	a, b, c, d	4	d, b, c, a	Exchange 1st and 4th stack items
0x93	SWAP4	5	$a_1..a_5$	5	a_5, $a_2..a_4$, a_1	Exchange 1st and 5th stack items
0x94	SWAP5	6	$a_1..a_6$	6	a_6, $a_2..a_5$, a_1	Exchange 1st and 6th stack items
0x95	SWAP6	7	$a_1..a_7$	7	a_7, $a_2..a_6$, a_1	Exchange 1st and 7th stack items
0x96	SWAP7	8	$a_1..a_8$	8	a_8, $a_2..a_7$, a_1	Exchange 1st and 8th stack items
0x97	SWAP8	9	$a_1..a_9$	9	a_9, $a_2..a_8$, a_1	Exchange 1st and 9th stack items
0x98	SWAP9	10	$a_1..a_{10}$	10	a_{10},$a_2..a_9$,a_1	Exchange 1st and 10th stack items
0x99	SWAP10	11	$a_1..a_{11}$	11	a_{11},$a_2..a_{10}$,a_1	Exchange 1st and 11th stack items
0x9A	SWAP11	12	$a_1..a_{12}$	12	a_{12},$a_2..a_{11}$,a_1	Exchange 1st and 12th stack items
0x9B	SWAP12	13	$a_1..a_{13}$	13	a_{13},$a_2..a_{12}$,a_1	Exchange 1st and 13th stack items
0x9C	SWAP13	14	$a_1..a_{14}$	14	a_{14},$a_2..a_{13}$,a_1	Exchange 1st and 14th stack items
0x9D	SWAP14	15	$a_1..a_{15}$	15	a_{15},$a_2..a_{14}$,a_1	Exchange 1st and 15th stack items
0x9E	SWAP15	16	$a_1..a_{16}$	16	a_{16},$a_2..a_{15}$,a_1	Exchange 1st and 16th stack items
0x9F	SWAP16	17	$a_1..a_{17}$	17	a_{17},$a_2..a_{16}$,a_1	Exchange 1st and 17th stack items

Logging Operations (0xA0–0xA4)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0xA0	LOG0	2	offset, size	0	—	Emit log with no topics
0xA1	LOG1	3	offset, size, $topic_1$	0	—	Emit log with 1 topic
0xA2	LOG2	4	offset, size, $topic_1$, $topic_2$	0	—	Emit log with 2 topics
0xA3	LOG3	5	offset, size, $topic_1$, $topic_2$, $topic_3$	0	—	Emit log with 3 topics
0xA4	LOG4	6	offset, size, $topic_1$..$topic_4$	0	—	Emit log with 4 topics (maximum)

System Operations (0xF0–0xFF)

Hex	Mnemonic	#In	Stack In	#Out	Stack Out	Description
0xF0	CREATE	3	value, offset, size	1	addr (0=fail)	Create new contract; push new address or 0 on failure
0xF1	CALL	7	gas, addr, value, argsOff, argsSize, retOff, retSize	1	success (0/1)	Message-call into another account
0xF2	CALLCODE	7	gas, addr, value, argsOff, argsSize, retOff, retSize	1	success (0/1)	Call with current storage but external code (deprecated)
0xF3	RETURN	2	offset, size	0	—	Halt and return mem[offset..offset+size] as output
0xF4	DELEGATECALL	6	gas, addr, argsOff, argsSize, retOff, retSize	1	success (0/1)	Like CALL but preserves msg.sender and msg.value (EIP-7)
0xF5	CREATE2	4	value, offset, size, salt	1	addr (0=fail)	Create contract at deterministic address (EIP-1014)
0xFA	STATICCALL	6	gas, addr, argsOff, argsSize, retOff, retSize	1	success (0/1)	Like CALL but reverts on state modification (EIP-214)
0xFD	REVERT	2	offset, size	0	—	Halt, revert state changes, return mem[offset..offset+size]
0xFE	INVALID	0	—	0	—	Designated invalid instruction; consumes all remaining gas
0xFF	SELFDESTRUCT	1	addr	0	—	Send balance to addr and mark contract for deletion (EIP-6049: deprecated)