Push data	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_pushboolean		L	bool			0		+1	-1: bool	-	push a boolean value
lua_pushinteger		L	n			0		+1	-1: num	-	push integer
lua_pushnumber		L	n			0		+1	-1: num	-	push Lua number (double)
lua_pushliteral		L	pc			0		+1	-1: string	-	push string literal
lua_pushstring		L	pc			0		+1	-1: string	-	push C string
lua_pushlstring		L	pc	len		0		+1	-1: string	m	push string of given length
lua_pushfstring	pc	L	fmt	…		0		+1	-1: string	m	push a sprintf() formatted string and return also pointer to result
lua_pushvfstring	pc	L	fmt	va_list		0		+1	-1: data	m	push a vsprintf() formatted string and return also pointer to result
lua_pushcfunction		L	cf			0		+1	-1: func	m	push a C function
lua_pushcclosure		L	cf	n		-n	-1: upvalue 1	+1	-1: func	m	push a C closure with n upvalues (value 1 being pushed first)
							-n: upvalue n				
lua_pushlightuserdata		L	data			0		+1	-1: data	-	push light user data
lua_pushthread		L				0		+1	-1: thread	-	push the thread of the current stack, return 1 for main thread
lua_pushnil		L				0		+1	-1: nil	-	push nil

Check data	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_isboolean	ok	L	n			0	-n: value	0		-	return true if value at position n is a boolean
lua_isnumber	ok	L	n			0	-n: value	0		-	return true if value at position n is a number
lua_isstring	ok	L	n			0	-n: value	0		-	return true if value at position n is a string
lua_istable	ok	L	n			0	-n: value	0		-	return true if value at position n is a table
lua_isfunction	ok	L	n			0	-n: value	0		-	return true if value at position n is a Lua function
lua_iscfunction	ok	L	n			0	-n: value	0		-	return true if value at position n is a C function
lua_islightuserdata	ok	L	n			0	-n: value	0		-	return true if value at position n is a light user data
lua_isuserdata	ok	L	n			0	-n: value	0		-	return true if value at position n is a light/full user data
lua_isoneornil	ok	L	n			0	-n: value	0		-	return true if value at position n is nil or outside current stack
lua_isnone	ok	L	n			0	-n: value	0		-	return true if value at position n is outside current stack
lua_isnil	ok	L	n			0	-n: value	0		-	return true if value at position n is nil
lua_isthread	ok	L	n			0	-n: value	0		-	return true if value at position n is a thread

Get data checked	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
luaL_checkany		L	n			0	-n: value	0		v	check if value at position n is a valid value
luaL_checkinteger	integer	L	n			0	-n: value	0		v	check for integer and return int at position n
luaL_checkint	int	L	n			0	-n: value	0		v	check for number and return int at position n
luaL_checklong	long	L	n			0	-n: value	0		v	check for number and return long at position n
luaL_checkstring	pc	L	n			0	-n: value	0		v	check for string and return string at position n
luaL_checklstring	pc	L	n	plen		0	-n: value	0		v	check for string and return string at position n and actual length
luaL_checknumber	number	L	n			0	-n: value	0		v	check for number and return int at position n
luaL_checktype		L	n	t		0	-n: value	0		v	check for Lua type t at position n
luaL_checkudata	data	L	n	name		0	-n: value	0		v	check for userdata name and return its pointer at position n
luaL_checkoption	index	L	n	pc	ppc	0	-n: value	0		v	search index of n (or pc) in list ppc

Get data converted	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_toboolean	bool	L	n			0	-n: value	0		-	convert value at position n to bool
lua_tocfunction	cf	L	n			0	-n: value	0		-	convert value at position n to a C function
lua_tointeger	int	L	n			0	-n: value	0		-	convert value at position n to integer
lua_tostring	pc	L	n			0	-n: value	0		-	convert value at position n to C string, return pointer
lua_tolstring	pc	L	n	plen		0	-n: value	0		-	convert value at position n to C string, return pointer and actual length
lua_tonumber	number	L	n			0	-n: value	0		-	convert value at position n to a Lua number
lua_topointer	data	L	n			0	-n: value	0		-	convert value at position n to pointer
lua_tothread	thread	L	n			0	-n: value	0		-	convert value at position n to thread
lua_touserdata	ud	L	n			0	-n: value	0		-	convert value at position n to light userdata

Get data with defaults	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
luaL_optint	int	L	n	d		0	-n: value	0		m	check for number at position n, return n if number or d otherwise
luaL_optinteger	integer	L	n	d		0	-n: value	0		m	check for number at position n, return n if number or d otherwise
luaL_optlong	long	L	n	d		0	-n: value	0		m	check for number at position n, return n if number or d otherwise
luaL_optnumber	number	L	n	d		0	-n: value	0		m	check for number at position n, return n if integer or d otherwise
luaL_optlstring	string	L	n	pc	len	0	-n: value	0		m	check for string at position n, return n if string or pc with len otherwise
luaL_optstring	string	L	n	pc		0	-n: value	0		m	check for string at position n, return n if string or pc otherwise

Stack operator	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_gettop	size	L				0		0		-	return the current size of the stack
lua_settop		L	n			?		?		-	set stack size to n
lua_insert		L	n			-1	-1: value	+1	-(n-1): value	-	moves top element to position n
lua_pop		L	n			-n		0		-	pop n values from stack
lua_pushvalue		L	n			0	-n: value	+1	-1: value	-	push value at position n
lua_remove		L	n			-1	-n: value	0		-	remove value at position n
lua_replace		L	n			-1	-1: value	0	-(n-1): value	-	pop value and replace value at position n
lua_xmove		L1	L2	n		-n		+n		-	pop n values from L1, push to L2
Value operator	**R**	**P1**	**P2**	**P3**	**P4**	**POP**	**Stack IN**	**PUSH**	**Stack OUT**	**Error**	**Remark**
lua_equal	eq	L	n1	n2		0	-n1: value 1	0		e	return true if values at position n1 and n2 are equal
							-n2: value 2				
lua_lessthan	eq	L	n1	n2		0	-n1: value 1			e	return true if value at position n1 is smaller than value at position n2
							-n2: value 2				
lua_rawequal	eq	L	n1	n2		0	-n1: value 1	0	0 x	-	return true if value at position n1 is smaller than value at position n2 (without metamethods)
							-n2: value 2				
luaL_gsub		L	pc	patt	rep	0		+1	-1: string	m	push copy of pc with all patt replaced by rep
lua_concat			n			-n	-1: value 1	+1	-1: string		pop n values and push concatenated strings 1 .. n
							-n: value n				

Table	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_createtable		L	narr	nrec		0		+1	-1: table	m	create and push a new table with pre-allocated space
lua_newtable		L				0		+1	-1: table	m	create and push a new empty table
lua_getfield		L	n	name		0	-n: table	+1	-1: value	e	push value of table at position n with field name
lua_setfield		L	n	name		-1	-1: key			e	pop value and store in table at position n with field key
							-n: table				
lua_rawget		L	n			-1	-1 field	+1	-1 value	e	push value of table at position n with field at top (without metamethods)
							-n: table				
lua_rawset		L	n			-2	-1: value	0		e	pop value and store in table at position n with field key (without metamethods)
							-2: key				
							-n: table				
lua_rawgeti		L	n	k		0	-n: table	+1	-1: value	-	push value of table at position n with field at index k
lua_rawseti		L	n	k		-1	-1: value				pop value and store in table at position n with field at index k
							-n: table				
lua_gettable		L	n			-1	-1: field	+1	-1: value	e	push value of table at position n with field at top
							-n: table				
lua_settable		L	n			-2	-1: value	0		e	pop value and store in table at position n with field key
							-2: key				
							-n: table				
lua_getmetatable	ok	L	n			0	-1: value	+0, +1	-1: table		push metatable of value at position n if possible
lua_setmetatable		L	n			-1	-1: value	0		-	pop value and store as metatable for value at position n
							-n: table				
LuaL_newmetatable	created	L	name			0		+1	-1: table	m	push metatable name in registry, create if necessary
lualL_getmetatable		L	name			0		+1	-1: table	-	push metatable name from registry
lua_next	ok	L	n			-1	-1: key	+0, +2	-1: value	e	pop key and push next pair of table at position n, (push nil at start, pop 1 value in each iteration
									-2: key		
lua_objlen	size	L	n			0	-n: value	0		-	return size of value at position n
luaL_getmetafield	ok	L	n	name		0	-n: value	+0, +1	-1: value	v	push metafield of value at position n with given name if possible

Global data	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_setglobal		L	name			-1	-1 value	0		e	store a global value
lua_getglobal		L	name			0		+1	-1: value	e	push a global value
lua_setfenv		L	n			-1	-1: table	0		-	set environment table of value at position n
							-n: table				
lua_getfenv		L	n			0		+1	-1: env	-	push environment table of value at position n
lua_register		L	name	func		0		0		-	register C function in global table with name
luaL_register		L	name	list		-0, -1	-1: table	+1	-1: table	m	open library with list elements in table on stack, if name is != 0 create and push new table

Call function	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_call		L	ni	no		-(ni+1)	-1: value ni	+no	-1: value 1	e	call Lua function func with ni input values and no expected return values
							-2: value 1		-n: value n		
							-(ni+1): func				
lua_pcall	error	L	ni	no	lf	-(ni+1)	-1: value ni	+no, +1	-1: value | errobj	-	call Lua function func in protected mode, if f != 0 call function at position f
							-2: value 1		-n: value n		
							-(ni+1): func				
lua_cpcall	error		cf	ud		0	-1: ud	+0, +1	-1: errobj	-	call C function in protected mode, if error != 0 return errobj
luaL_callmeta	ok	L	n	name		0	-n: value	0, +1	-1: value	e	call metatable of value at position n with field name if possible: v = n.__call(n)

Load or call Lua code	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_load	error	L	reader	data	name	0		+1	-1: func	-	load a Lua chunk name by repeatedly calling reader(data), push the resulting function
luaL_loadbuffer	error	L	pc	len	name	0		+1	-1: func	m	load a Lua chunk with name at given buffer and length, push the resulting function
luaL_dofile	ok	L	file			0		?	-?: value ?	m	load and run Lua file, push return values
luaL_dostring	ok	L	pc			0		?	-?: value ?	m	load and run Lua chunk in memory, push return values
luaL_loadfile	error	L	file			0		+1	-1: func	m	load a Lua file, push the resulting function
luaL_loadstring	error	L	pc			0		+1	-1: func	m	load a Lua chunk in memory, push the resulting function

Debugging	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_gethook	cf	L				0		0		-	return hook function
lua_gethookcount	n	L				0		0		-	return hook count
lua_gethookmask	n	L				0		0		-	return hook mask
lua_sethook		L	cf	mask	count	0		0		-	set hook function, mask and count
lua_getinfo	n	L	what	ar		0, -1	-1: func	+(0,1,2)		m	return specific information, **see manual for details**
lua_getlocal	name	L	ar	n		0		+0, +1	-1: value	-	get information for local variable
lua_setlocal	name	L	ar	n		-0, -1	-1: value	0		-	pop and store as value of local variable n
lua_getupvalue	name	L	f	n		0		+0, +1	-1: value	-	get information for upvalue n in function at position f
lua_setupvalue	name	L	f	n		0, -1	-1: func	0		-	pop and store as value of upvalue n in function at position f
lua_dump	error	L	writer	data		0	-1: func	0		-	dump function as binary data to writer
lua_error		L				-1	-1: message	0		-	generates a Lua error, never returns
lua_getstack	ok	L	n	ar		0		0		-	get information of runtime stack at level n
lua_checkstack	n	L	n			0		0		m	ensure remaining stack space of at least n values
lua_type	t	L	n			0	-n: value	0		-	return Lua type of value at position n
lua_typename	name	L	t			0		0		-	return typename of type number t
lua_atpanic	cf	fn				0		0		-	set panic function and return previous one
luaL_argcheck		L	cond	n	msg	0		0		v	If cond is not true: raise argument error with text based on n and msg
luaL_argerror		L	n	msg		0		0		v	raise argument error with text based on n and msg
luaL_typerror		L	n	name		0		0		-	raise type error with text based on n and name
luaL_error		L	fmt	…		0		0		m	raise error with a sprintf() formatted message
luaL_checkstack		L	n	msg		0		0		m	ensure remaining stack space of at least n values, raise error with text including msg
luaL_where		L	n			0		+1	-1: string	m	push a string describing the current programm position

Buffer	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
luaL_buffinit		L	B			0		0		-	initialise a buffer
luaL_prepbuffer	data	B				0		0		-	return intermediate space
luaL_addvalue		B				-1	-1: value	0		m	pop value and copy resulting string to buffer
luaL_addchar		B	c			0		0		m	add character c to buffer
luaL_addlstring		B	pc	len		0		0		m	add string with len l to buffer
luaL_addstring		B	pc			0		0		m	add C string to buffer
luaL_addsize		B	n			0		0		m	add intermediate space with given size to buffer
luaL_pushresult		B				0		+1	-1: value	m	finish buffer and push result

Thread	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_yield		L	n			-n		+n		-	suspend a coroutine
lua_resume	error	L	n			-n		+n		-	resume a coroutine
lua_status	error	L				0		0		-	return status of thread L

Library	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_close		L				0		0		-	close Lua library
luaopen_base		L				0		0		-	open base library
luaopen_debug		L				0		0		-	open debug library
luaopen_io		L				0		0		-	open io library
luaopen_math		L				0		0		-	open math library
luaopen_os		L				0		0		-	open os library
luaopen_package		L				0		0		-	open package library
luaopen_string		L				0		0		-	open string library
luaopen_table		L				0		0		-	open table library
luaL_openlibs		L				0		0		-	open all the above standard libraries

Misc	R	P1	P2	P3	P4	POP	Stack IN	PUSH	Stack OUT	Error	Remark
lua_newthread		L				0		+1	-1: table	m	create and push a new thread
lua_newuserdata		L	size			0		+1	-1: table	m	allocate and push user data with given size
lua_newstate	state	alloc	ud			0		0		-	create a new Lua state with given allocator and user data
luaL_newstate	state					0		0		-	create a new Lua state with defaults
lua_gc	error	L	what	data		0		0		e	control garbage collector
lua_getallocf	alloc	L	pud			0		0		-	get memory allocator and user data
lua_setallocf		L	func	ud		0		0		-	set memory allocator and user data
luaL_ref	key	L	n			-1	-1: value	0		m	create unique key in table at position n
							-n: table				
luaL_unref		L	n	key		0	-n: table	0		-	release key in table at position n

luaL_findtable											deprecated: ?
luaL_setn											deprecated: set table size, see table.setn()
luaL_getn											deprecated: get table size, see table.getn()
luaL_openlib											open library

deprecated function
values popped in call
values remaining in call

Basic types	Value	Type name
LUA_TNONE	(-1)	
LUA_TNIL	0	nil
LUA_TBOOLEAN	1	boolean
LUA_TLIGHTUSERDATA	2	
LUA_TNUMBER	3	number
LUA_TSTRING	4	string
LUA_TTABLE	5	table
LUA_TFUNCTION	6	function
LUA_TUSERDATA	7	
LUA_TTHREAD	8	thread

Thread status	
LUA_YIELD	1
LUA_ERRRUN	2
LUA_ERRSYNTAX	3
LUA_ERRMEM	4
LUA_ERRERR	5

Pseudo indices	
LUA_REGISTRYINDEX	(-10000)
LUA_ENVIRONINDEX	(-10001)
LUA_GLOBALSINDEX	(-10002)
lua_upvalueindex(i)	(LUA_GLOBALSINDEX-(i))

Name	Remark
cf	c function
lf	Lua function
ud	light userdata
ni	number input args
no	number output args
k	field index
n	Stack element at index n
n1	Stack element 1 at index n1
n2	Stack element 2 at index n2
value	Lua value
fmt	format string, see fprintf()
data	pointer to raw data
number	Lua number
ar	Pointer to debug structure
L, L1, L2	Lua state
B	Lua buffer
integer	Lua integer
t	lua type
ok	1=success
error	error code, 0=ok