LuaByteSharp

Kurzbeschreibung des Projekts

LuaByteSharp soll ein Interpreter für Lua Byte-Code werden. Der Interpreter soll in C# geschrieben werden.
Das Programm soll fähig sein, den vom Lua Standardcompiler luac, in der Version 5.3, erzeugten Byte-Code einzulesen und auszuführen.
Geplant ist, dass alle der 47 existierenden Operationen (siehe) unterstützt werden.
Als Quellen für die Beschreibung der Opertationen, sowie des generellen Formats eines LuaByteCode Files, werden die unter Quellen gelisteten Quellen verwendet. Geplant ist ebenfalls, dass das Metatable-Konzept (siehe) von Lua unterstützt wird.

Da Lua-Code oft auf der Lua Standard Library aufbaut, soll zumindest ein Subset der Standard Library unterstützt werden.
Das Subset wird wahrscheinlich je nach verfügbarer Zeit angepasst werden. Jedenfalls werden aber weder `require` noch `load` bzw. `loadfile` und infolgedessen auch nicht `dofile` untersützt werden, da alle diese Funktionen Lua Sourcecode laden können, der Interpreter diesen aber nicht parsen kann.

Außerdem muss das Lua Byte Code File kompatibel zur 64-Bit Version von luac sein, und im Little-Endian Format vorliegen. Es muss folgender Header vorhanden sein:

```
00:     1B 4C 75 61 53 00 19 93
08:     0D 0A 1A 0A 04 08 04 08
10:     08 78 56 00 00 00 00 00
18:     00 00 00 00 00 00 28 77
20:     40
```

Detailierte Auflistung der Bestandteile

Byte Code Instruktionen

siehe hier
bzw. hier

Standard Library Funktionen

siehe auch

Basic Functions

Bis auf die folgenende Ausnahmen, sollen alle Basic Functions unterstützt werden:

- `collectgarbage`
- `dofile`
- `load`
- `loadfile`
- `pcall`
- `xpcall`

String Manipulation

Auch hier ist geplant, bis auf einige Ausnahmen, alle Funktionen zu unterstützen.
Ausnahmen:

- `string.dump`
- `string.pack`
- `string.packsize`
- `string.unpack`

UTF-8 Support

Die folgenden Methoden sollen alle unterstützt werden:

- `utf8.char`
- `utf8.charpattern`
- `utf8.codes`
- `utf8.codepoint`
- `utf8.len`
- `utf8.offset`

Table Manipulation

Die folgenden Methoden sollen alle unterstützt werden:

- `table.concat`
- `table.insert`
- `table.move`
- `table.pack`
- `table.remove`
- `table.sort`
- `table.unpack`

Mathematical Functions

Alle mathematischen Funktionen der Lua Standard Library sollen unterstützt werden.

Mögliche Erweiterungen

Möglicherweise werden auch alle oder Teile der IO Funktionen unterstützt werden.
Auch die Funktionen `string.pack`, `string.packsize` und `string.unpack` werden vielleicht unterstützt.

Aufwandsabschätzungen

Komponente	Aufwand
Interpreter Grundgerüst	10 h
Byte Code Instruktionen	20 h
Standard Library	40 h

Byte Code Instruktionen

Gruppe	Instruktionen	Aufwand
Move/Load	MOVE, LOADK, LOADKX, LOADBOOL, LOADNIL	2 h
Up-Value/Table	(GET/SET)UPVAL, (GET/SET)TABUP, (GET/SET)TABLE, NEWTABLE	2 h
Arithmetik	ADD, SUB, MUL, DIV, IDIV, MOD, POW, UNM	1 h
Bit Manipulation	BAND, BOR, BXOR, SHL, SHR, BNOT	1 h
Logik	TEST, TESTSET, NOT	1 h
Vergleiche	EQ, LT, LE	1 h
Schleifen	FORLOOP, FORPREP, TFORLOOP, TFORCALL	2 h
Funktionen	CALL, TAILCALL, RETURN, CLOSURE	6 h
Andere	JMP, VARARG, SETLIST, CONCAT, LEN, SELF, EXTRAARG	4 h

Standard Library Funktionen

Gruppe	Funktionen	Aufwand
Basis Funktionen	assert, error, _G, getmetatable, ipairs, next, pairs, print, rawequal, rawget, rawlen, rawset, select, tonumber, tostring, type, _VERSION	10 h
String Manipulation	string.byte, string.char, string.find, string.format, string.gmatch, string.gsub, string.len, string.lower, string. match, string.rep, string.reverse, string.sub, string.upper	15 h
Mathematik	math.abs, math.acos, math.asin, math.atan, math.ceil, math.cos, math.deg, math.exp, math.floor, math.fmod, math.huge, math.log, math.max, math.maxinteger, math.min, math.mininteger, math.modf, math.pi, math.rad, math.random, math.randomseed, math.sin, math.sqrt, math.tan, math.tointeger, math.type, math.ult	5 h
Table Manipulation	table.concat, table.insert, table.move, table.pack, table.remove, table.pack, table.remove, table.sort, table.unpack	5 h
UTF-8 Support	utf8.char, utf8.charpattern, utf8.codes, utf8.codepoint, utf8.len, utf8.offset	5 h

Quellen

Lua Byte Code File Format

- http://luaforge.net/docman/83/98/ANoFrillsIntroToLua51VMInstructions.pdf (nur teilweise)
- http://files.catwell.info/misc/mirror/lua-5.2-bytecode-vm-dirk-laurie/lua52vm.html (nur teilweise)
- https://www.lua.org/source/5.3/ldump.h.html
- https://www.lua.org/source/5.3/ldump.c.html
- https://www.lua.org/source/5.3/lundump.h.html
- https://www.lua.org/source/5.3/lundump.c.html

Byte Code Instruktionen

- https://www.lua.org/source/5.3/lopcodes.h.html
- https://www.lua.org/source/5.3/lvm.c.html
- https://github.com/dibyendumajumdar/ravi/blob/master/readthedocs/lua_bytecode_reference.rst

Standard Library Funktionen

- https://www.lua.org/manual/5.3/manual.html#6
- https://www.lua.org/source/5.3/lbaselib.c.html
- https://www.lua.org/source/5.3/lstrlib.c.html