Compiler Design for Lua

Synopsis Report

Submitted by

- Aritra Basu - 160905126 - Roll no. 17
- Pavan Kalyan - 160905138 - Roll no. 22
- Sai Vignesh - 160905142 - Roll no. 23

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Institution of Manipal University

Introduction

Lua is an extension programming language designed to support general procedural programming with data description facilities.
It also offers good support for object-oriented programming, functional programming, and data-driven programming.
Lua is intended to be used as a powerful, light-weight scripting language for any program that needs one. Lua is implemented as a library, written in clean C (that is, in the common subset of ANSI C and C++).

BNF Grammar for Lua

```
chunk ::= {stat [";"]} [laststat [";"]]

block ::= chunk

stat ::=  varlist "=" explist |
 functioncall |
 do block end |
 while exp do block end |
 repeat block until exp |
 if exp then block {elseif exp then block} [else block] end |
 for Name "=" exp "," exp ["," exp] do block end |
 for namelist in explist do block end |
 function funcname funcbody |
 local function Name funcbody |
 local namelist ["=" explist]

laststat ::= return [explist] | break

funcname ::= Name {"." Name} [":" Name]

varlist ::= var {"," var}

var ::=  Name | prefixexp "[" exp "]" | prefixexp "." Name

namelist ::= Name {"," Name}

explist ::= {exp ","} exp

exp ::=  nil | false | true | Number | String | "..." | function |
```

```
 prefixexp | tableconstructor | exp binop exp | unop exp

prefixexp ::= var | functioncall | "(" exp ")"

functioncall ::=  prefixexp args | prefixexp ":" Name args

args ::=  "(" [explist] ")" | tableconstructor | String

function ::= function funcbody

funcbody ::= "(" [parlist] ")" block end

parlist ::= namelist ["," "..."] | "..."

tableconstructor ::= "{" [fieldlist] "}"

fieldlist ::= field {fieldsep field} [fieldsep]

field ::= "[" exp "]" "=" exp | Name "=" exp | exp

fieldsep ::= "," | ";"

binop ::= "+" | "-" | "*" | "/" | "^" | "%" | ".." |
 "<" | "<=" | ">" | ">=" | "==" | "~=" |
 and | or

unop ::= "-" | not | "#"
```