```
//* Copyright 2019 Rochus Keller <mailto:me@rochus-keller.ch>
//*
//* This file is part of the LjAsm parser library.
//*
//* The following is the license that applies to this copy of the
//* library. For a license to use the library under conditions
//* other than those described here, please email to me@rochus-keller.ch.
//*
//* GNU General Public License Usage
//* This file may be used under the terms of the GNU General Public
//* License (GPL) versions 2.0 or 3.0 as published by the Free Software
//* Foundation and appearing in the file LICENSE.GPL included in
//* the packaging of this file. Please review the following information
//* to ensure GNU General Public Licensing requirements will be met:
//* http://www.fsf.org/licensing/licenses/info/GPLv2.html and
//* http://www.gnu.org/copyleft/gpl.html.

LjAsm ::= function_decl

function_decl ::=  function [ fname ] function_header [ function_body ] end [ fname ]

function_header ::=  '(' formal_params ')'
      [ const_decls ] [ var_decls ]  { function_decl }

function_body ::=  begin  { [ labelDef ] statement }

labelDef ::= label ':'

formal_params ::=  { vname }

var_decls ::= var { var_decl | record }

var_decl ::= vname

record ::= '{' vname { vname } '}' // the fields occupy consecutive slots and are in
                                   // the same namspace as the other vars

const_decls ::= const  cname '=' const_val { cname '=' const_val  }

const_val ::= string | number | primitive | table_literal

table_literal ::= '{' { [ vname '=' ] ( string | number | primitive ) } '}'

desig ::= [  fname '.' ] // optional function name prefix for upvalues
      vname

statement ::= ISLT_ |  ISGE_ | ISLE_ | ISGT_ | ISEQ_ | ISNE_
      | ISTC_ | ISFC_ | IST_ | ISF_ | MOV_ | NOT_ | UNM_ | LEN_
      | ADD_ | SUB_ | MUL_ | DIV_ | MOD_
      | POW_ | CAT_ | KSET_ | KNIL_ | UGET_ | USET_
      | UCLO_ | FNEW_ | TNEW_ | TDUP_ | GGET_ | GSET_ | TGET_ | TSET_
      | CALL_ | CALLT_ | RET_ | FORI_ | FORL_ | LOOP_ | JMP_

ISLT_ ::= ISLT desig desig
ISGE_ ::= ISGE desig desig
ISLE_ ::= ISLE desig desig
ISGT_ ::= ISGT desig desig

ISEQ_ ::= ISEQ desig ( desig | string | number | primitive )
```

// = ISEQV, ISEQS, ISEQN and ISEQP, desig includes cname

ISNE_ ::= ISNE desig **(** desig **|** string **|** number **|** primitive **)**
// = ISNEV, ISNES, ISNEN and ISNEP, desig includes cname

ISTC_ ::= ISTC desig desig
ISFC_ ::= ISFC desig desig
IST_ ::= IST desig
ISF_ ::= ISF desig

MOV_ ::= MOV desig desig

NOT_ ::= NOT desig desig
UNM_ ::= UNM desig desig

LEN_ ::= LEN desig desig

ADD_ ::= ADD desig **(** desig **|** number **) (** desig **|** number **)**
SUB_ ::= SUB desig **(** desig **|** number **) (** desig **|** number **)**
MUL_ ::= MUL desig **(** desig **|** number **) (** desig **|** number **)**
DIV_ ::= DIV desig **(** desig **|** number **) (** desig **|** number **)**
MOD_ ::= MOD desig **(** desig **|** number **) (** desig **|** number **)**
// = xVN, xNV and xVV, desig includes cname

POW_ ::= POW desig desig desig
CAT_ ::= CAT desig desig **[** posint **]** // posint: number of slots to cat, all to end of record/array if left out

KSET_ ::= KSET desig **(** string **|** number **|** primitive **|** cname **)**
// = KSTR, KCDATA, KSHORT, KNUM and KPRI
KNIL_ ::= KNIL desig **[** posint **]** // posint: number of slots to cat, all to end of record/array if left out

UGET_ ::= UGET desig desig // dst <- uv
USET_ ::= USET desig **(** string **|** number **|** primitive **|** desig **)** // uv <- src
// = USETV, USETS, USETN and USETP, desig includes cname

UCLO_ ::= UCLO desig **[** label **]**
FNEW_ ::= FNEW desig fname

TNEW_ ::= TNEW desig **[** posint **[** posint **]]** // optional array size and hash size
TDUP_ ::= TDUP desig **(** cname **|** table_literal **)**
GGET_ ::= GGET desig **(** string **|** cname **)**
GSET_ ::= GSET desig **(** string **|** cname **)**
TGET_ ::= TGET desig desig **(** desig **|** string **|** posint **)**
// = TGETV, TGETS and TGETB; desig includes cname
TSET_ ::= TSET desig desig **(** desig **|** string **|** posint **)**
// = TSETV, TSETS and TSETB; desig includes cname

CALL_ ::= CALL desig **[** posint **[** posint **]]** // number of return values, number of arguments
CALLT_ ::= CALLT desig **[** posint **]** // number of arguments

RET_ ::= RET [desig **[** posint **]]** // base slot and number of slots to return
// = RET, RET0, RET1

FORI_ ::= FORI desig label
FORL_ ::= FORL desig label
LOOP_ ::= LOOP
JMP_ ::= JMP label

primitive ::= nil | true | false

label ::= ident // jump label name
vname ::= ident // variable name
cname ::= ident // constant name
fname ::= ident // function name

number ::= real | integer
integer ::= negint | posint

ident ::=
string ::=
real ::=
posint ::=
negint ::=