Avaliação prática: implementação do *front-end* de um compilador

Compiladores

2016.2

1 Definição

Implementar um programa que recebe como entrada um código em linguagem Lua (`http://lua.org`) e retorna a tradução para um código de três endereços. A linguagem Lua está descrita na EBNF abaixo.

⟨*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
| '#'

2 Da implementação

O parser pode utilizar tanto a abordagem *top-down* como *bottom-up*. Podem, ainda, ser utilizadas as ferramentas Flex e Bison (bem como suas interfaces para outras linguagens). O programa deve ser

feito em C, C++, Java, Lua, Python, Haskel e/ou Lisp. Demais linguagens de programação devem ser consultadas previamente. Demais ferramentas devem ser consultadas previamente.

3 Entregáveis

A entrega do trabalho constará de:

- código fonte (com Makefile ou roteiro completo de compilação/execução) do programa,
- relatório descrevendo o processo de construção do *software* (i.e. técnicas de compilação utilizadas e tabelas de *lookahead* e/ou *Action/GoTo* quando apropriado),
- arquivos de exemplo e
- apresentação do trabalho.

A entrega e agendamento da apresentação devem ser feitos por email até o dia 21 de dezembro de 2016. A apresentação do trabalho deve ser feita até o dia 4 de janeiro de 2017.