Lua i LuaJIT

- Stefan Pejić
- RT-RK, Computer Based Systems
- Fakultet tehničkih nauka, Novi Sad
- Današnje teme su:
  - Programski jezik Lua
  - Programski prevodilac LuaJIT

Glavne osobine jezika

- Portabilnost
- Jednostavnost
- Mala veličina
- Skriptovanje

Portabilnost

- Podrška za raznovrsne operativne sisteme i procesorske arhitekture
- Veoma lako se prilagođava novim arhitekturama
- Napisan u skladu sa ANSI C standardom
- LuaJIT nema ove osobine

    Koristi asembler, noviji C standard i raznovrsne GCC ekstenzije

Jednostavnost

- Jednostavna sintaksa
- Mala standardna biblioteka
- Dinamički tipiziran jezik
    - Nema eksplicitnih tipova
- Interpretiran
    - Nema potrebe za prevođenjem

Skriptovanje

- Proširivanje aplikacija napisanih u nekom drugom (obično sistemskom) jeziku
- Lua je implentirana kao biblioteka
- Proširivanje aplikacija napisanih u Lui nekim drugim programskim jezikom
- Dosta modula (biblioteka) za Luu je napisano u C-u

Osobine Lue

```
local math = require("math")  -- Ucitavanje modula

local x = 5.4                 -- Lokalna promjenljiva
y = math.floor(x)             -- Globalna primjenljiva

local z = {                   -- Niz/mapa
  "jabuka",
  "kruska",
  elem1 = 12,
  elem2 = y
}
print(z.elem1)                -- Pristup elementu mape
print(z[1])                   -- Pristup elementu niza

for key,value in pairs(z) do  -- Iterator
  print(key, value)
end

for i = 1, 100 do             -- Numericka for petlja
  x = x + i
end
```

Upotreba Lue

- Veliki broj frameworka za pisanje igara
  - LÖVE, Corona, MOAI, itd.
  - Stingray 3D, Luxinia, Cafu 3D, itd.
- Igre proširive korišćenjem Lue:
  - The Witcher
  - Far Cry
  - World of Warcraft
  - Google „Lua-scripted video games“

Upotreba Lue

- Adobe Photoshop Lightroom
- VLC media player
- Apache web server
- Wikipedia
- Cisco Adaptive Security Appliance
- I mnogi drugi

Kako početi?

- Knjiga „Programming in Lua“, Roberto Ierusalimschy
- Lua zajednica: www.lua-users.org
- Mailing lista: http://www.lua.org/lua-l.html
- Lua priručnik: http://www.lua.org/manual/5.3/

Kako početi?

- Postoji više implementacija Lue:
- Lua (zvanična implementacija) i LuaJIT su najpoznatiji kompajleri za Luu
- http://www.lua.org/download.html
- http://luajit.org/download.html

LuaJIT

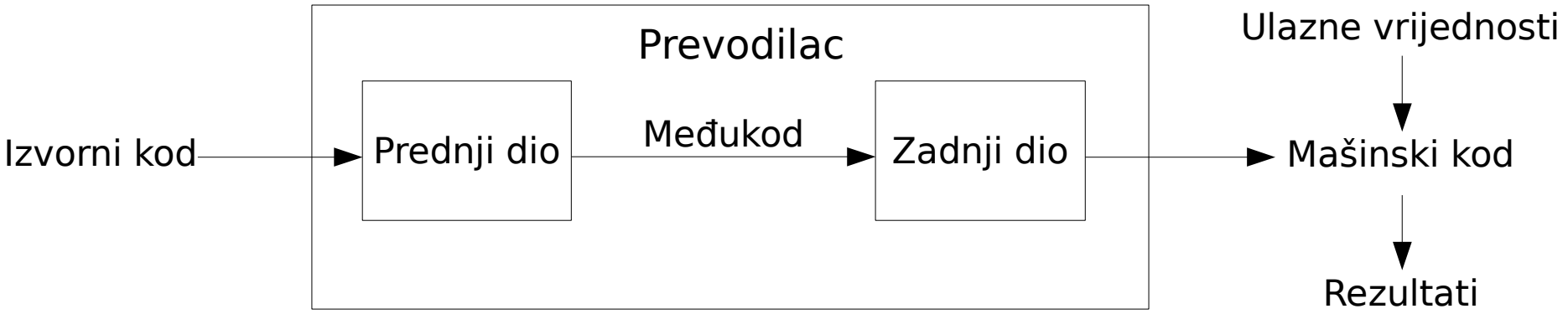
- Znatno brže izvršavanje koda napisanog u Lui
- Nudi ubrzanje od 2 do 100 puta
- Mala upotreba memorije
- Mala izvršna datoteka
- Dolazi sa dodatnim modulima
- FFI modul
- Bit modul

LuaJIT

- Loša portabilnost
- Jako je teško podržati novu arhitekturu
- Podržane arhitekture: x86, x64, ARM, ARM64, MIPS, MIPS64, PPC
- Ne podržava nove verzije Lue
- Trenutno kompatibilan sa verzijom 5.1

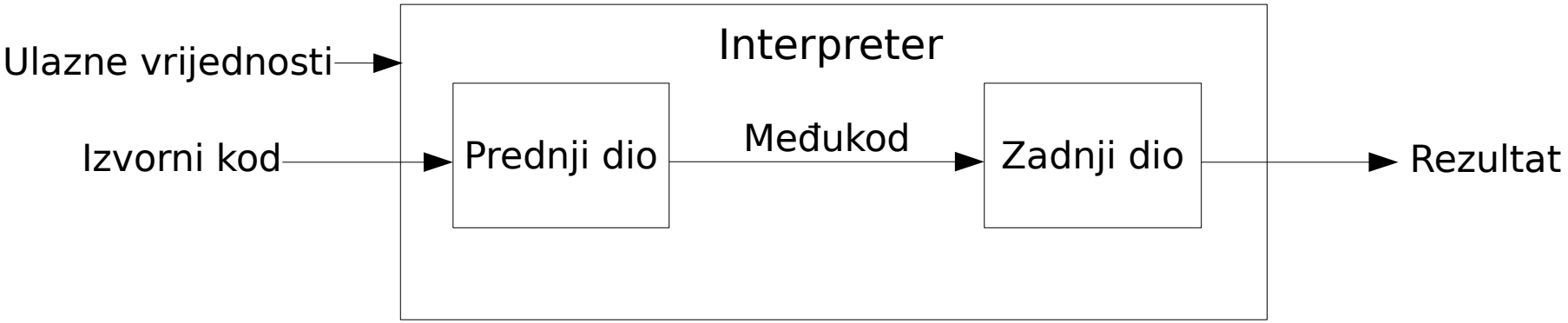
Programski prevodilac

- Prevodi izvorni kod u mašinski kod
- Obično se sastoji iz dva dijela
- Prednji analizira izvorni i generiše međukod
- Zadnji optimizuje međukod i generiše mašinski kod



Interpreter

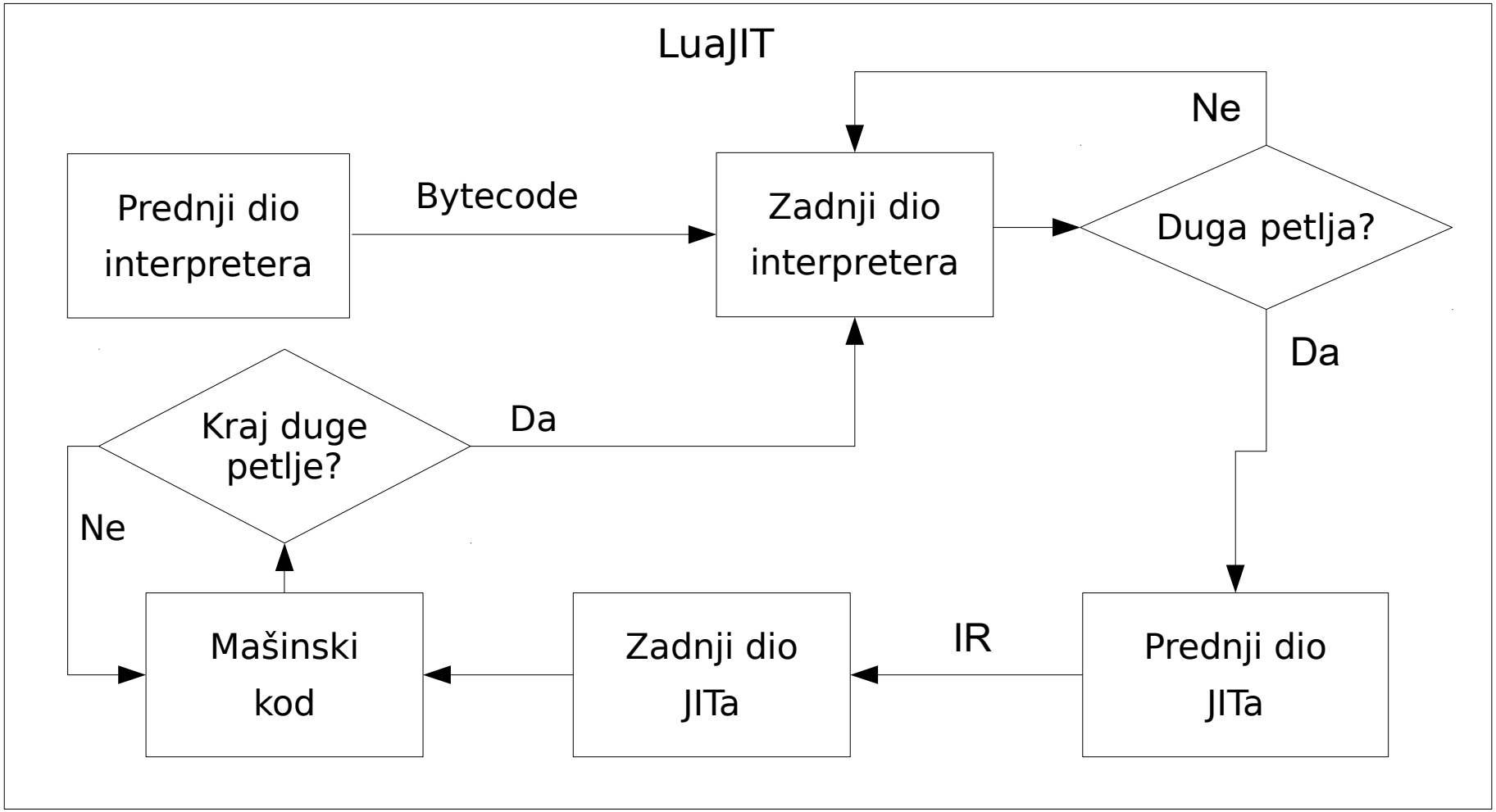
- Program koji izvršava izvorni kod
- Dosta sličnosti sa klasičnim prevodiocem
- Izlaz nije mašinski kod
- Prednji dio je slican kao kod prevodioca
- Zadnji dio izvršava dobijeni međukod



LuaJIT

- Interpreter + JIT prevodilac
- Kod prvo prolazi kroz interpreter koji ga izvršava
- Vrši se analiza pri interpretiranju
- Detektuju se petlje u kojima se program najduže zadržava
- Te petlje se proslijeđuju JIT prevodiocu koji ih dodatno optimizuje i generiše mašinski kod
- Mašinski kod se zatim izvršava

LuaJIT


LuaJIT
Prednji dio interpretera
Bytecode
Zadnji dio interpretera
Duga petlja?
Ne
Da
Kraj duge petlje?
Da
Ne
Mašinski kod
Zadnji dio JITa
IR
Prednji dio JITa

Interpreter LuaJITa

- Napisan u asembleru
- Interpretira međukod (bytecode)
- Međukod je pažljivo dizajniran za moderne procesore
- 4-bajtne instrukcije
- Lako dostupni operandi

Interpreter LuaJITa

Lua code:

```
-----------------------
local min = 100
local max = 1000
local write = io.write

for i = min, max do
  write("Hello!")
end
```

Bytecode:

```
-------------------------------------
0001    KSHORT   0 100
0002    KSHORT   1 1000
0003    GGET     2   0      ; "io"
0004    TGETS    2   2   1  ; "write"
0005    MOV      3   0
0006    MOV      4   1
0007    KSHORT   5   1
0008    FORI     3 => 0013
0009 => MOV      7   2
0010    KSTR     8   2      ; "Hello!"
0011    CALL     7   1   2
0012    FORL     3 => 0009
0013 => RET0     0   1
```

JIT prevodilac

- Napisan u programskom jeziku C
- Generiše mašinski kod
- Neportabilan
- Radi nad drugom vrstom međukoda (IR kod)
- Bytecode interpretera se transformiše u IR kod
- IR kod je povoljniji za JIT jer sadrži više informacija

IR kod

Lua code:

```
-----------------------
local x = 5

for i = 1, 100 do
  X = x + i
end
```

IR code:

```
-------------------------------
0001     int SLOAD  #2    CI
0002 >   num SLOAD  #1    T
0003     num CONV   0001  num.int
0004   + num ADD    0003  0002
0005   + int ADD    0001  +1
0006 >   int LE     0005  +100
0007 ------ LOOP ------------
0008     num CONV   0005  num.int
0009   + num ADD    0008  0004
0010   + int ADD    0005  +1
0011 >   int LE     0010  +100
0012     int PHI    0005  0010
0013     num PHI    0004  0009
```

Prilagođavanje LuaJITa

- Prilagođavanje interpretera
- Prilagođavanje JIT prevodioca
- Potrebno upoznavanje sa ciljnom arhitekturom
- Registri
- Instrukcije
- Konvencije

Prilagođavanje interpretera

- 99% asembler, 1% C
- Potrebno poznavanje međukoda interpretera
- Svodi se na implementiranje svih instrukcija međukoda i nekih pomoćnih funkcija
- Ručno biranje registara
- Poređenje sa postojećim implementacijama za drugi arhitekture
- Pronalaženje efikasnih načina za implementaciju kompleksnih operacija

Prilagođavanje interpretera

- Implementaciju instrukcije MOV u interpreteru

```
|.define BASE,   x19    // Base of current Lua stack frame.
|.define RA,     x27
|.define RC,     x28
|.define TMP0,   x8

case BC_MOV:
  |  // RA = dst, RC = src
  |  ldr TMP0, [BASE, RC, lsl #3]
  |  str TMP0, [BASE, RA, lsl #3]
  |  ins_next
  break;
```

Prilagođavanje JITa

- 85% C, 15% asembler
- Upoznavanje sa IR kodom
- Upoznavanje sa alokatorom registara
- Kod se generiše unazad
- Kod je specijalizovan (ima tipove) i baziran na pretpostavkama
- Pretpostavke moraju biti tačne, a u slučaju da nisu, mora se vratiti u interpreter

Prilagođavanje JITa

- Funkcija koja generise masinski kod za određivanje minimuma ili maksimuma

```
static void asm_intmin_max(ASMState *as, IRIns *ir, A64CC cc)
{
  Reg dest = ra_dest(as, ir, RSET_GPR);
  Reg left = ra_hintalloc(as, ir->op1, dest, RSET_GPR);
  Reg right = ra_alloc1(as, ir->op2, rset_exclude(RSET_GPR, left));
  emit_dnm(as, A64I_CSELw|A64F_CC(cc), dest, left, right);
  emit_nm(as, A64I_CMPw, left, right);
}
```

Stefan Pejić – stefan.pejic@rt-rk.com

RT-RK, Computer Based Systems

www.rt-rk.com