Domaći zadatak

Sistemsko programiranje IR3SP

Septembar 2012

Nemanja Trifunović 346/2010

Zadatak [40p]

Zadatak se sastoji od tri dela: izrada interpretativnog emulatora za arhitekturu opisanu u prilogu, izrada JIT emulatora za istu arhitekturu i poređenje performansi implementiranih emulatora.

a) (15 poena) Izraditi interpretativni emulator za arhitekturu opisanu u prilogu. Emulator na ulazu prihvata apsolutizovan predmetni program bez ikakvih dodatnih informacija, nakon čega pokreće i izvršava program. Pri emulaciji in i out instrukcija koristiti standardni ulaz i izlaz.

b) (25 poena) Izraditi JIT emulator za arhitekturu opisanu u prilogu. Ulaz u emulator, kao i način emulacije in i out instrukcija je identičan kao u slučaju interpretativnog emulatora. Pri tome, jednom izgenerisani kod nije potrebno menjati (nije potrebna optimizacija da se u slučaju već generisanog koda zaobiđe povratak u glavnu petlju emulatora). Da bi se pojednostavilo prevođenje, na raspolaganju je kompajler TinyCC koji kroz biblioteku "libtcc.a" pruža podršku za prevođenje u runtime-u. Ova biblioteka ima mogućnost da fragment C koda zadat kao string prevede u mašinski kod koji je već učitan u memoriju i spreman za izvršavanje. Od studenata se očekuje da delove mašinskog koda emulirane mašine prevedu u odgovarajući C kod, a da generisani C kod dalje prevode korišćenjem pomenute biblioteke. Delovi koda koji se prevode treba da budu bazični blokovi.

c) Pripremiti bar 3 testa za prethodna dva emulatora. Testovi treba da pokriju sve funkcionalnosti emulatora, ali i da budu tako napravljeni da je vreme izvršavanja tih testova dovoljno dugo da se može meriti. Kao deo rešenja, potrebno je pokrenuti pripremljene testove na oba emulatora, i za svaki emulator i svaki pripremljeni test, dati izmereno vreme izvršavanja. Kao vreme izvršavanja uzeti srednje vreme izvršavanja testa u 3 uzastopna pokretanja.

Rešenje

Rešenje problema je dato u prilogu. Prilog se sastoji od foldera *sp* sa kodom rešenja, skriptama za prevođenje rešenja, pomoćnim skriptama i primerima programa za emuliranu mašinu.

Ulazna tačka za interpretativni emulator se nalazi u fajlu *interpreter.cpp*, a za jit emulator u fajlu *jit.cpp*.

Da bi se zadatak preveo potrebno je iskopirati folder *sp* i skripte za prevođenje (date u prilogu) u folder *tcc-0.9.25*. Prevodjenje programa se postiže pokretanjem odgovarajuće skripte. Skripti se opciono mogu proslediti parametri koje će kompajler koristiti pri prevođenju.

Skripte za prevođenje:

<skripta>	**-->**	**<izlazni program>**
prevedi	-->	emulator
prevedi_jit	-->	jit_emulator
prevedi_writeProgram	-->	writeProgram

Lista parametra koji se mogu proslediti skripti za prevodjenje:

- -D LOG – ispisuje log emulatora na standardnom izlazu
- -D LOG1 – opširniji log
- -D LOG2 – još opširniji log
- -D LOG3 – najopširniji log
- -D LOGFILE = lokacija_log_fajla – preusmerava log u zadati fajl
- -D DEBUG – log-uje stanje emulirane mašine posle svake instrukcije
- (*) -D SELF_MODIFYING – omogućuje emulaciju samomodifikujućeg koda

primer korišćenja: ./prevedi_jit -D SELF_MODIFYING -D LOG

* Samo kod JIT emulatora (obican emulator uvek podržava ovu opciju)

Primeri su dati u izvršnom (.izv) i pseudo izvršnom formatu (ovaj format dozvoljava komentare). Program *writeProgram* konvertuje pseudo izvršni u izvršni format. Program se prevodi skriptom *prevedi_writeProgram*. U prilogu postoje pomoćne skripte *_emulator* i *_jit_emulator* koje pripremaju izvršni fajl i pozivaju emulator ili jit_emulator respektivno.

Primeri dati u prilogu:

- primer01 – učitava dva broja sa standardnog ulaza, sabira ih i ispisuje rešenje na standardni izlaz
- primer02 – rekurzivno poziva potprogram za učitavanje broja sa standarnog ulaza dok se ne unese 0
- primer03 – učitava broj iz memorije, rekurzivno ga smanjuje i ispisuje na standardni izlaz dok ne dodje do 0
- (*) primer04 – rekurzivno računa fibonačijev broj
- primer05 – samomodifikujući program
- (*) primer06 – vrši obradu nad nizovima
- (*) primer07 – vrši obradu nad nizovima

Pomocne skripte

- _emulator
- _jit_emulator

primer korišćenja: ./emulator primer01.izv

primer korišćenja: ./_emulator primer01

* Primeri korišćeni za poređenje performansi emulatora

Rezultati uporednog merenja performansi

program	emulator	jit_emulator
program04 [param 40]	1m24.160s	1m11.104s
program06	0m1.948s	0m1.444s
program07	0m10.144s	0m6.840s

Zaključak: Kao što je bilo i očekivano JIT emulator je pokazao bolje performanse od iterativnog emulatora.