Elektrotehnički fakultet Univerziteta u Beogradu



Konstrukcija emulatora za asembler

– Domaći zadatak iz predmeta Sistemsko Programiranje –

|  |  |
| --- | --- |
| Student: |  |
| Vladimir Sladojević 2014/48 |

Beograd, Avgust 2018.

**Postavka zadatka**

Napraviti interpretativni emulator za računar opisan u prilogu. Ulaz emulatora je izlaz asemblera, pri čemu je moguće zadati veći broj predmetnih programa koje je potrebno učitati, povezati i pokrenuti. Nazivi predmetnih programa se zadaju kao argumenti komandne linije. Emulacija je moguća samo ukoliko nakon učitavanja i povezivanja nema preklapanja između učitanih fajlova, nema nerazrešenih simbola niti višestrukih definicija simbola i postoji definisan simbol START koji definiše početnu adresu za izvršavanje.

**Opis rešenja**

Rešenje je realizovano na Ubuntu operativnom sistemu u programu c++, korišćeni standard je c++11.

Kao što pomenuto u postavci, argumenti programa su predmetni fajlovi koji su dobijeni kao izlaz asemblera. Prva faza rešenja se sastoji iz čitanja tih predmethih fajlova i vršenja provera, da li dolazi do preklapanja bilo koja dva fajla, da li su svi simboli definisani, da li je definisan ulazni simbol START. Nakon toga, vrši se obrada zapisa o relokacijama. Sve ovo u cilju da bi nakon provera, uspešnog povezivanja fajlova dobili konačan sadržaj memorije koji prosleđujemo klasi Emulator koja vrši dalju obradu.

Glavna funkcija klase Emulator, funkcija emulate, funkcioniše na jednostavnom principu. Izvršava istu petlju sve dok je uslov za rad emulatora aktivan. Sama petlja radi sledeće: dohvata instrukciju sa sadržaja na koji ukazuje pc, vrši dekodovanje instrukcije, izvršavanje instrukcije, a potom obradu prekida.

Osim funkcije emulate, aktivne su još dve niti. Prva nit izvršava funkciju keyboard, a druga printListener. Sistem dakle funkcioniše na sledeći način: izvršava se funkcija emulate koja dohvata instrukcije i izvršava ih; nit keyboard čeka da korisnik unese karakter na tastaturi i onda reaguje upisujući pročitani karakter na adresu ulazne periferije; nit printListener osluškuje da li je potrebno ispisati sadržaj sa adrese izlazne periferije. Niti keyboard i printListener kada se „probude“, osim što vrše upis/čitanje, postavljaju zahtev za prekidom. Zahtevi sa prekidom se obrađuju u funkciji handleInterrupt koja u slučaju da prekid nije u toku, a da je generisan zahtev za prekid, poziva odgovarajuću prekidnu rutinu, koju čita iz odgovarajućeg ulaza IVT tabele. U tabeli se nalaze adrese prekidnih rutina keyboardInterrupt i timerInterrupt koje su napisane u nekom od asemblerskih fajlova, čiji su predmetni programi prosleđen kao argumenti samog programa.

**Uputstvo za prevođenje i pokretanje**

Za prevođenje i pokretanje se koristi *gcc* kompajler , verzija 4.9.

Takođe, potrebno je instalirati biblioteku *“ncurses.h”* komandom *apt-get install libncurses5-dev*, ukoliko već ne postoji na mašini.

Prevođenje se vrši sledećom komandom:

*g++ -std=c++11 -o emulator Emulator.cpp main.cpp Emulator.h Structures.h -lpthread -lncurses* ,

a pokretanje:

*./emulator OUTA.txt OUTB.tx*t .

**Testovi**

