Elektrotehnički fakultet Univerziteta u Beogradu



Konstrukcija dvoprolaznog asemblera

– Domaći zadatak iz predmeta Sistemsko Programiranje –

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

Beograd, Avgust 2018.

**Postavka zadatka**

Napisati dvoprolazni asembler za procesor opisan u tekstu zadatka. Ulaz asemblera je tekstualni fajl koji je u skladu sa sintaksom opisanom u tekstu zadatka. Izlaz asemblera treba da bude predmetni program baziran na školskoj varijanti elf formata zapisan u tekstualnom fajlu.

**Opis rešenja**

Rešenje je realizovano na Ubuntu operativnom sistemu, a korišćen je programski jezik c++.

Kao što je pomenuto u postavci, ulazni fajl predstavlja kod napisan u asembleru (čije pojedinosti su opisane u tekstu postavke zadatka). Cilj je od tog asemblerskog koda napraviti fajl koji predstavlja predmetni program, a koji sadrži u sebi: tabelu simbola, sadržaje sekcija (heksadecimalne reprezentacije instrukcija) koje se javljaju u ulaznom fajlu, kao i zapise o relokacijama. Prema tome, neophodno je ulazni fajl „čitati i analizirati“, kako bi se najpre ustanovilo da li je fajl ispravno napisan, a potom ako jeste, treba napraviti pomenuti izlazni fajl.

Da bi se ovo ostvarilo, u rešenju se koristi princip dvoprolaznog asemblera. U prvom prolazu formiramo „nekompletnu“ tabelu simbola, a u drugom prolazu konačnu tabelu simbola, sadržaje sekcija i zapise o relokacijama, čime smo kompletirali sadržaj izlaznog fajla, koji potom ispisujemo u sam fajl.

U prvom prolazu prolazimo kroz ulazni fajl, obrađujući liniju po liniju. Uklanjamo komentare, odsecamo bele znakove sa početka i sa kraja linije, ako je nakon toga sadržaj linije prazan, preskačemo red. U suprotnom, proveravamo da li obrađena linija sadrži labelu, predstavlja sekciju. Potom proveravamo da li predstavlja direktivu ili instrukciju. Za potrebe ovih provera, napisani su regularni izrazi koji predstavljaju dozvoljen format direktiva i instrukcija koje asembler treba da prepozna. Dakle, nakon početne obrade linije, vrši se poređenje sa svakim od regularnih izraza. Ako se ne pronađe poklapanje, javlja se greška sa porukom i brojem linije na kojoj je greška registrovana. U slučaju poklapanja, od sadržaja linije se kreira objekat klase Instruction koji sadrži sve neophodne informacije za drugi prolaz. Zahvaljujući tome, nakon prvog prolaza možemo zatvoriti ulazni fajl za čitanje jer nema potrebe da za drugi prolaz ponovo prolazimo kroz fajl i ponovo čitamo sve linije i vršimo neophodne potrebe. Svi kreirani objekti klase Instruction se nalaze u listi. Osim toga, poenta prvog prolaza je da registrujemo sve sekcije i labele koje se javljaju i dodamo ih u tabelu simbola.

Drugi prolaz se sastoji iz iteriranja kroz pomenutu listu. Sada se proverava da li su neki tipovi adresiranja nedozvoljeni za određene instrukcije, u kom slučaju se javlja poruka o grešci. Ako su provere uspešne, vrši se dekodovanje instrukcije i ako je potrebno, za instrukciju se kreira zapis o relokaciji. Takođe, u drugom prolazu se obrađuju direktive .global na osnovu kojih se vrše eventualne promene u tabeli simbola.

Nakon toga, sadržaj koji predstavlja predmetni program se ispisuje u tekstualni fajl zadatak kao drugi parametar programa. Bitno je napomenuti da prvi parametar predstavlja put do ulaznog tekstualnog fajla, a treći parametar predstavlja adresu početka smeštanja programa. (Drugi i treći parametar su opcioni)

**Uputstvo za prevođenje i pokretanje**

Za prevođenje i pokretanje se koristi *gcc* kompajler. Prema tome, ukoliko već ne postoji na samoj Ubuntu mašini, neophodno je instalirati g++ verziju 4.9.

Prevođenje se vrši komandom:

*g++ -std=c++11 -o asembler main.cpp* ,

a pokretanje komandom:

*./asembler putanjaUlaznogFajla.txt {putanjaIzlaznogFajla.txt adresapočetka}* .

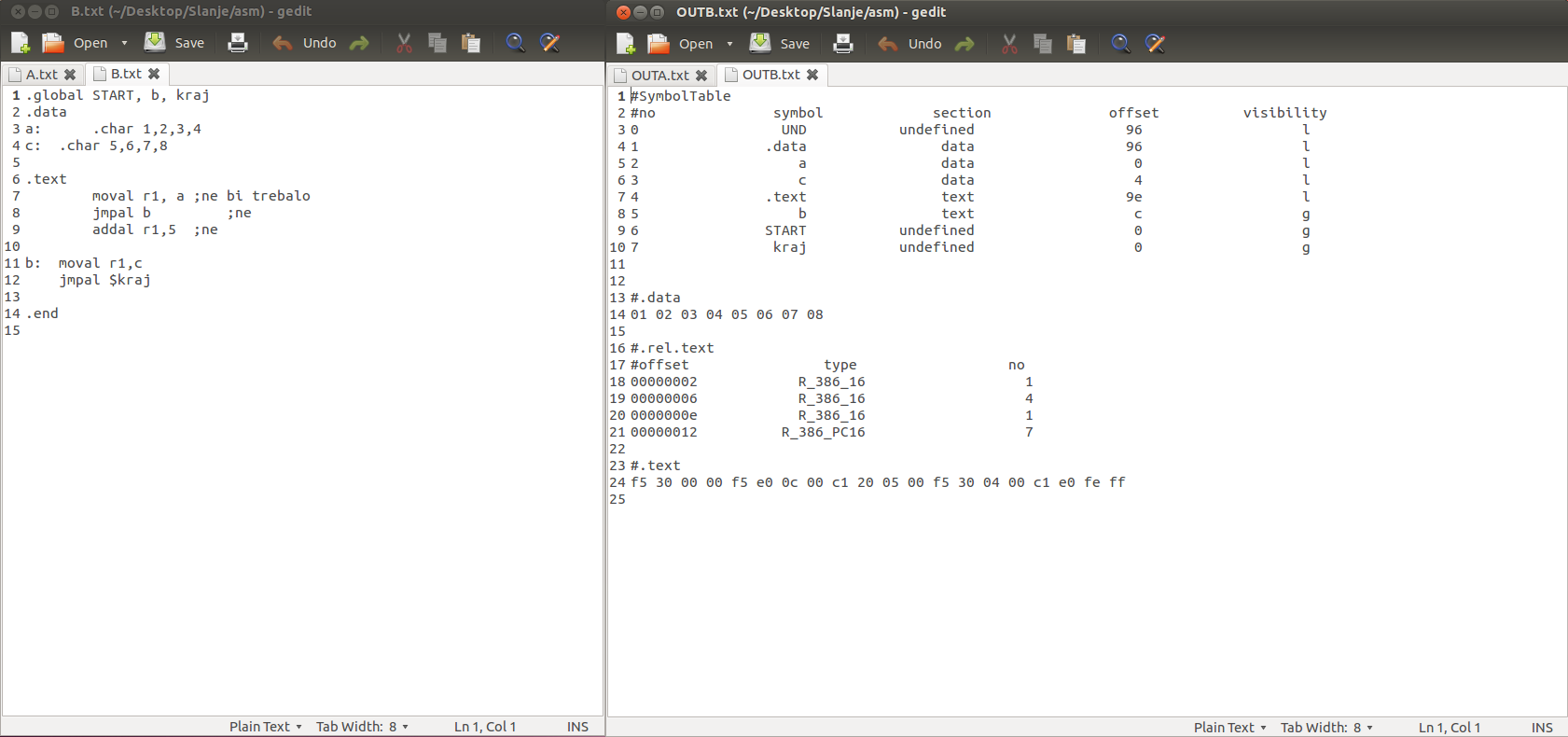
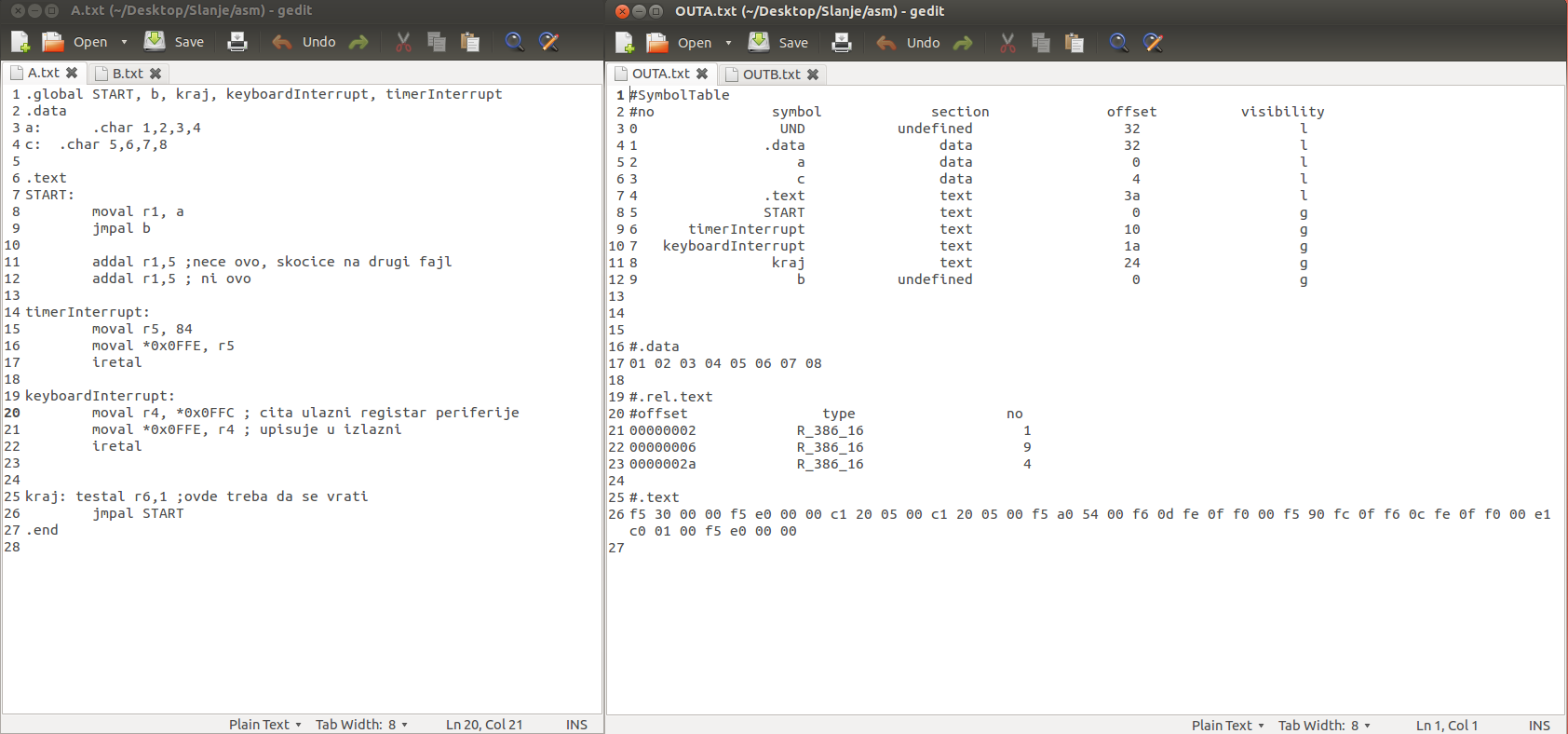
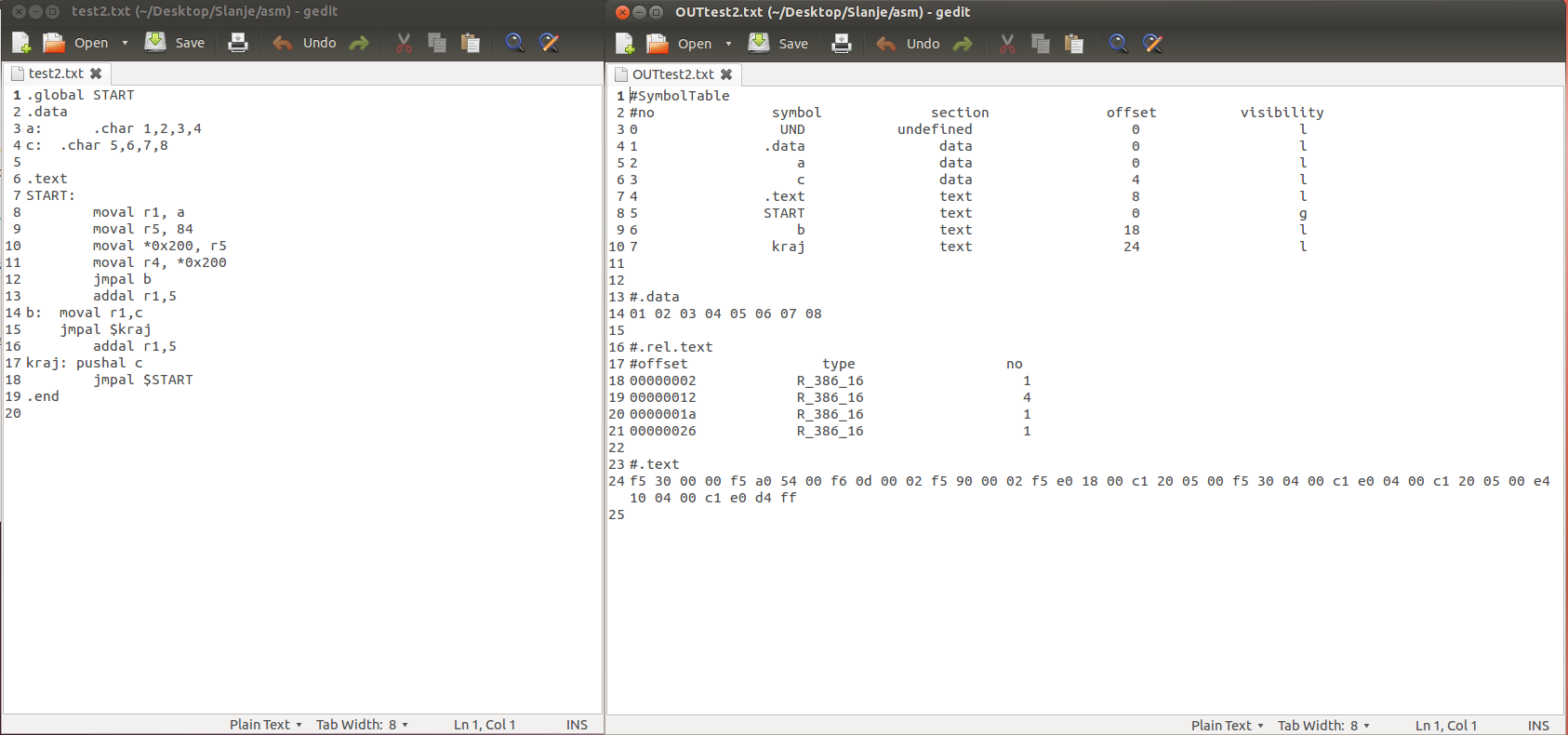
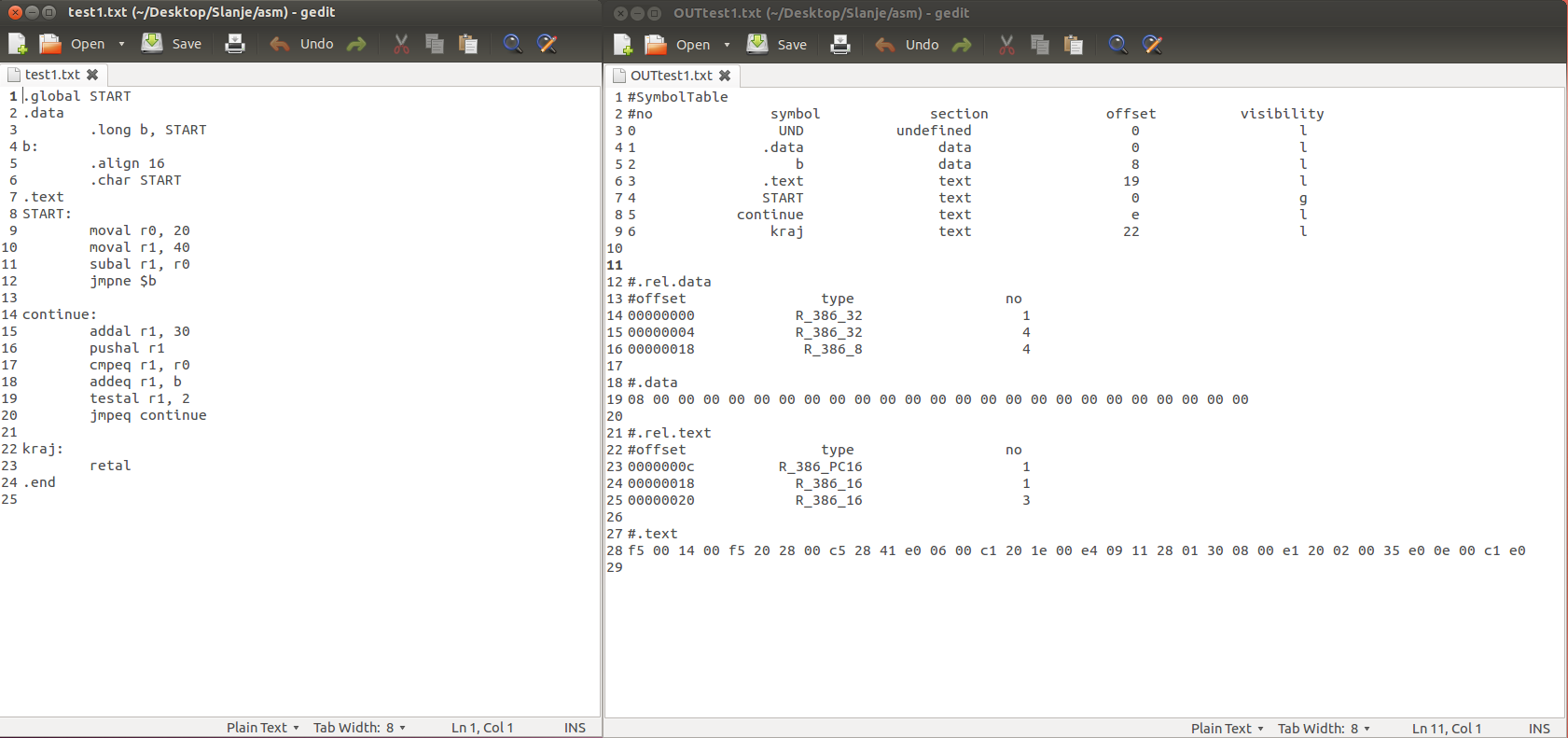
Primeri dozvoljenih komandi za pokretanje:

*./asembler test.txt*

*/.asembler test.txt OUT-test.txt*

*/.asembler test.txt OUT-test.txt 10* .

**Testovi**

****