Elektrotehnički fakultet Beograd

Sistemski softver(13E113SS)

**Konstrukcija jednoprolaznog asemblera**

**Student: Ana Vanzo 2015/0029**

**Postavka zadataka:**

Potrebno je konstruistati jednoprolazni asembler za računarski sistem koji je priložen uz tekst projektnog zadatka. Ulaz asemblera je tekstualni fajl koji mora biti napisan u skladu sa zadatim sintaksnim pravilima. Izlaz asemblera treba da bude predmetni program koji je zapisan u tekstualnom fajlu i njegov format treba bazirati na školskoj varijanti elf formata.

**Opis rešenja:**

Rešenje je realizovano na Ubuntu 20.04 operativnom sistemu, na programskom jeziku c++ .

Asembler je realizovan u jednom prolazu. Program kroz komandnu liniju kao treći argument prima naziv tekstualnog fajla u kome se nalaze asemblerske naredbe. Prvi argument je “–o” iza koga sledi treći argument koji predstavlja naziv izlaznog tekstualnog fajla.

Program čita liniju po liniju ulaznog fajla sve dok ne pročita .end direktivu koja označava kraj asembliranja. Nakon pročitane linije određuje se tip pročitanog, da li je u pitanju labela, direktiva ili instrukcija. Pošto je u pitanju jednoprolazni asembler paralelno se popunjava tabela simbola i generiše odgovarajući kod, generiše se sadržaj sekcija i tabela relokacionih zapisa. Takođe, kako samo jedom prolazimo kroz ulazni fajl, potrebne su nam još neke dodatne strukture podataka koje će da reše glavni problem kod jednoprolaznog asemblera a to je obraćanje unapred. Kao pomoćna struktura podataka uvedena je tabela obraćanja unapred, koja za svaki simbol, kojem se unapred obraćamo, čuva njegovo ime i još listu struktura koje predstavljaju sva konkretna mesta upotrebe tog simbola u kodu. Ta struktura sadrzi sekciju u kojoj se koristi simbol, offset u toj sekciji, veličinu podatka (1B ili 2B) i tip relokacije ukoliko je to potrebno. Na kraju prvog i jedinog prolaza asemblera, iteriramo kroz tu tabelu obraćanja unapred, i razrešavamo sva ta mesta gde se koriste ti simboli. Od pomoćnih struktura podataka koristila sam još i tabelu equ simbola, tj. simbola definisanih .equ direktivom.

**Uputstvo za prevođenje:**

Prevođenje i pokretanje se vrši pomoću  *g++* kompajlera i terminala Ubuntu 20.04 operativnog sistema. Korišćena verzija gcc kao i g++ kompajlera je 9.30.

Kompajliranje se vrši tako što se svi fajlovi sa izvornim kodom stave u jedan isti folder, pa se u treminalu komandom cd postavi tekući direktorijum na folder sa izvornim kodom i zatim se izvrši sledeća komanda za kompajliranje:

**g++ –o nazivIzvršnogFajla ./\***

Pokretanje se vrši sledećom komandom:

**./nazivIzvršnogFajla nazivUlaznogFajla.s -o nazivIzlaznogFajla.s**

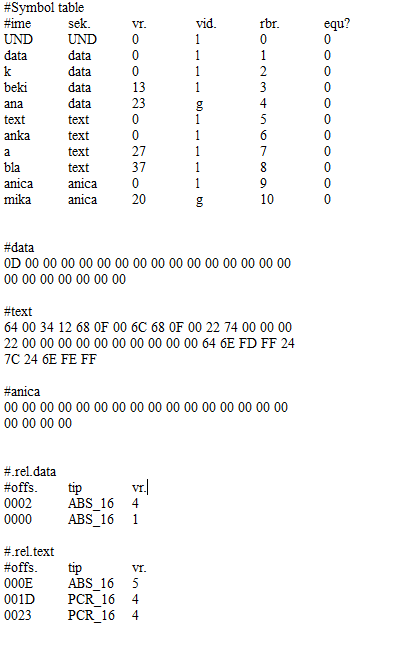
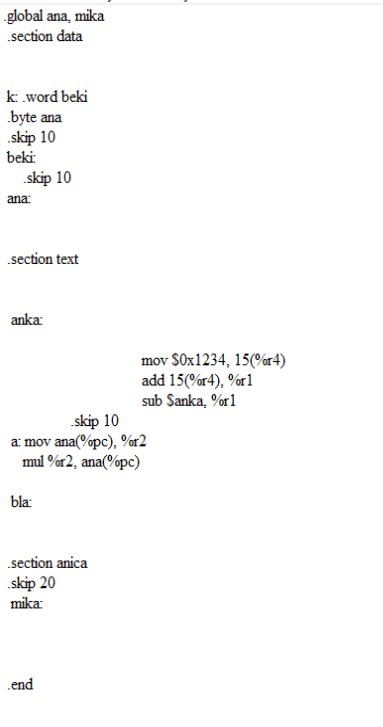
**Dodatne napomene:**

* Uvedena je pretpostavka da svi simboli iz izraza equ direktive moraju biti iz iste sekcije, takođe equ direktiva može da se koristi samo u data sekciji tj. sekciji definisanoj direktivom .section data, gde ja data naziv sekcije. S obzirom na to da je simbol definisan equ direktivom apsolutan, on ne bi trebalo da bude relokatibilan, tj vrednost equ simbola mora biti konstanta. Equ direktiva i služi da uglavnom definišemo neke konstante poput dužine bafera, broja elemenata niza itd.
* U okviru izraza equ direktive izmedju svakog simbola, literala i znaka +- mora postojati space, jer sam kao delimiter za tokenizaciju koristila space.
* U instrukcijama sa dva operanda uvedeno je ograničenje da ne mogu oba operanda biti memorijskog tipa, što je u skladu sa x86 asemblerom koji izučavali na predavanjima i vežbama.
* Veličina operanda, ukoliko to nije eksplicitno naglašeno odgovarajućim sufiksom u nazivu instrukcije, se određuje na osnovu veličine destinacionog operanda u instrukcijama sa 2 operanda, ukoliko je destinacioni operand memorijskog tipa određuje se po veličini izvorišnog operanda. U instrukcijama sa jednim operandom ukoliko je on memorijskog tipa, tj. njegova vrednost se nalazi u memoriji, usvojeno je da je njegova veličina 2B.
* Kod registarskih adresiranja ako je definisan sufiks h koduje se sa bitskom jedinicom, ako je sufiks l i u svim ostalim slučajevima bit L/H se koduje sa 0.
* Tipovi relokacionih zapisa označeni su sa ABS\_16 za apsolutnu relokaciju, i sa PCR\_16 za pc relativnu relokaciju
* .global i .extern direktive moraju biti korišćene isključivo na samom početku fajla
* Kod pc relativnog adresiranja simbola, ako je simbol definisan u istoj sekciji u kojoj se koristi, ne pravi se relokacioni zapis. Pomeraj koji treba da se upiše u kod same instrukcije kod pc relativnog adresiranja se dobija kao delta=simbol-PC, i neka je \* adresa u sekciji na koju treba da se upiše taj pomeraj (delta), tada se PC dobija kao \* + 2, pošto će u trenutku izvršavanja koda pc pokazivati na narednu instrukciju. Tako da se delta dobija kao simbol-\*-2. Međutim ukoliko se prvi operand, u instrukciji sa 2 operanda, adresira pc relativno, tada taj pomeraj (delta) iznosi simbol-\*-3, zbog toga što zasigurno znamo da je veličina Op2Descr u kodu instrukcije 1 B, zbog uvedenog ograničenja da ne mogu oba operanda biti memorijskog tipa, tako da drugi operand zasigurno mora biti registarski.

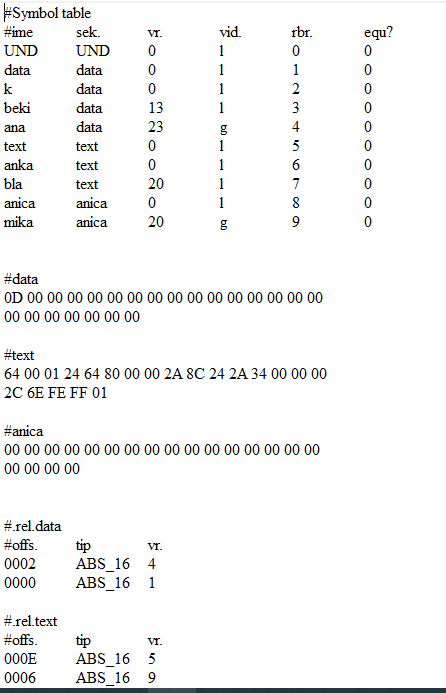
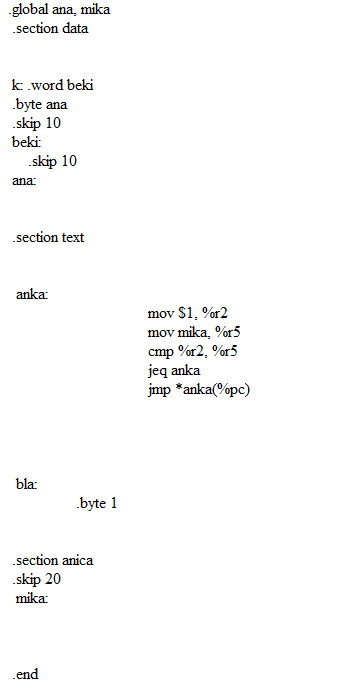
**Testovi:**

Dalje u prilogu se nalaze testovi sa njihovim asemblerskim kodom i izlazom implementiranog jednoprolaznog asemblera.

**Test 1.**

****

**Test 2.**

****

**Test 3.**

