SISTEMSKI SOFTVER 13E113SS, 13S113SS

PROJEKTNI ZADATAK POČEV OD JUNA 2020.

Napomene

Rok za predaju rešenja projektnog zadatka je nekoliko dana pred termin ispita. Način predaje rešenja i termin odbrane biće naknadno objavljeni. Rešenje se sastoji od sledećih stavki:

1) dokumentacija koja sadrži opis rešenja kao i uputstvo za prevođenje i pokretanje čitavog sistema,

2) testovi (ulazni fajlovi) koji prikazuju sve funkcionalnosti realizovanog sistema,

3) izvorni kod rešenja i

4) izvršni kod rešenja.

Odbrana projekta biće organizovana u svakom ispitnom roku. Obaveštenje o početku i pravilima predaje kao i tačan termin odbrane rešenja projektnog zadatka biće blagovremeno poslati na mejling listu predmeta (13e113ss@lists.etf.rs, 13s113ss@lists.etf.rs). Prilikom odbrane biće korišćena verzija rešenja projektnog zadatka koja je predata i neće biti dozvoljene naknadne izmene. Nepoštovanje pravila u vidu propusta u okviru dokumentacije i testova uzrokuje dobijanje manjeg broja poena.

Opis okruženja

Izrada rešenja projektnog zadatka radi se pod operativnim sistemom Linux na x86 arhitekturi u programskom jeziku C/C++. Potrebni alati su opisani u okviru materijala za predavanja i vežbe. Za one koji nemaju instaliran Linux ponuđena je instalacija u okviru virtuelne mašine VMware Player. Virtuelna mašina se može preuzeti sa internet stranice predmeta. U okviru ove virtuelne mašine već su instalirani potrebni alati. Odbrana rešenja projektnog zadatka vrši se isključivo pod prethodno opisanim okruženjem u okviru pomenute virtuelne mašine.

Zadatak I – Asembler (25 bodova)

Napisati jednoprolazni asembler za procesor opisan u prilogu. Svi potrebni argumenti asemblera (naziv ulaznog i izlaznog fajla) zadaju se kroz komandnu liniju prilikom pokretanja. Ulaz asemblera je tekstualni fajl u skladu sa sintaksom opisanom u nastavku. Za potrebe leksičke analize ulaznog tekstualnog fajla dozvoljeno je koristiti generatore leksera. Izlaz asemblera treba da bude predmetni program zapisan u tekstualnom fajlu. Za potrebe učitavanja u emulator dozvoljeno je generisati i binarni fajl pored tekstualnog.

Format predmetnog programa bazirati na školskoj varijanti ELF formata (tekstualni fajl kakav je korišćen u zadatku 9 u prezentaciji V3_Konstrukcija asemblera.ppt) i predložiti izmene u formatu u skladu sa potrebama ciljne arhitekture (sekcije, tipovi zapisa o relokacijama, dodatna polja u postojećim tipovima zapisa, novi podaci o predmetnom programu i slično).

Prilikom generisanja izlaza asemblera voditi se principima koje koristi GNU asembler. Sve sekcije smeštaju se počev od nulte adrese. Sintaksa asemblera i ostali zahtevi:

- jedna linija izvornog koda sadrži najviše jednu asemblersku naredbu/direktivu,

- labela, koja se završava dvotačkom, se može naći na početku linije izvornog koda nakon proizvoljnog broja belih znakova,

- labela može da stoji samostalno, bez prateće asemblerske naredbe/direktive u istoj liniji izvornog koda, što je ekvivalentno tome da stoji u liniji izvornog koda sa prvim sledećim sadržajem,

- simboli se izvoze navođenjem asemblerske direktive `.global <lista_simbola>` pri čemu je u okviru jedne direktive moguće navesti više simbola razdvojenih zapetama,

- simboli se uvoze navođenjem asemblerske direktive `.extern <lista_simbola>` pri čemu je u okviru jedne direktive moguće navesti više simbola razdvojenih zapetama,

- izvorni kod koji predstavlja asemblerske naredbe i asemblerske direktive za generisanje sadržaja moraju biti u okviru sekcija definisanih asemblerskom direktivom `.section <ime_sekcije>`:

- fajl sa izvornim kodom se završava (ostatak fajla se odbacuje tj. ne prevodi se) pomoću asemblerske direktive `.end`,

- alokacija prostora vrši se pomoću asemblerskih direktiva `.byte <lista_simbola/literala>`, `.word <lista_simbola/literala>` i `.skip <literal>` čije funkcionalnosti odgovaraju istoimenim asemblerskim direktivama GNU asemblera,

- definicija novih simbola, pored navođenja labela, moguća je pomoću asemblerske direktive `.equ <simbol>, <izraz>` gde `<izraz>` predstavlja sekvencu simbola i literala razdvojenih operatorima plus i minus,

- ostatak sintakse, ukoliko nije definisan u prilogu, definisati u skladu sa razumnim pretpostavkama.

Primer komande, kojom se pokreće asembliranje izvornog koda u okviru fajla *ulaz1.s*, dat je u nastavku:

```
asembler -o ulaz1.o ulaz1.s
```

Zadatak II – Interpretativni emulator (+15 poena)

Napisati interpretativni emulator za računarski sistem opisan u prilogu. Ulaz emulatora jeste proizvoljan broj fajlova koji predstavljaju izlaz asemblera. Nazivi ulaznih fajlova zadaju se kao argumenti komandne linije prilikom pokretanja emulatora. Emulacija je moguća samo ukoliko (1) ulazni predmetni programi nakon povezivanja mogu biti učitani u memorijski adresni prostor emuliranog računarskog sistema, (2) nema preklapanja između sekcija definisanih u okviru ulaznih predmetnih programa uzimajući u obzir `-place` argumente komandne linije i (3) prilikom povezivanja ulaznih predmetnih programa nema višestrukih definicija simbola niti nerazrešenih simbola.

Argumentom komandne linije `-place=<ime_sekcije>@<adresa>` definiše se adresa počev od koje se smešta sekcija navedenog imena. Argument komandne linije `-place` moguće je navesti veći broj puta, a sekcije za koje nije naveden smeštaju se proizvoljnim redom odmah iza sekcije koja je smeštena na najvišu adresu. Ukoliko u više ulaznih fajlova postoje sekcije istog imena njihov međusobni položaj u okviru istoimene agregirane sekcije je proizvoljan.

Primer komande, kojom se pokreće emulacija nad programom koji se dobija povezivanjem predmetnih programa *ulaz1.o* i *ulaz2.o* pri čemu se definišu adrese na koje se smeštaju odgovarajuće sekcije, dat je u nastavku:

```
emulator -place=iv_table@0x0000 -place=text@0x4000 ulaz1.o ulaz2.o
```