**Sistemski softver**

**Domaći zadatak**

Tijana Kušljević

Avgust 2018.

**Problem:**

Potrebno je napraviti dvoprolazni asembler koji za zadati ulazni tekstualni fajl pravi tabelu simola, relokacione zapise i izlaze sekcija koje upisuje u izlazni tekstualni fajl.

**Opis rešenja:**

Pri pokretanju programa u funkciji main se poziva kostruktor Asembler(), gde se za svaku moguću instrukciju generiše njen binarni operacioni kod na šest bita. Za povezivanje imena instrukcije i njenog koda koristi se klasa Sifra.

Za dva prolaza asemblera napisane su dve funkcije u klasi Asembler. Pri prvom prolazu se čita ceo fajl, koji je prosleđen kao prvi argument funkcije, do linije .end. Proverava se da li rec koja je nađena označava neku od sekcija. Postoje bss, data, rodata i text sekcija. Drugi argument funkcije je zadata adresa od koje se sekcije upisuju. Proverava se da li je bilo prethodne sekcije i koja je njena veličina da bi se znala početna adresa. Brojač se restartuje na nulu. Pravi se objekat klase simbol i popunjavaju se odgovarajuća polja. U mapi tabela čuvaju se svi simboli.

Ukoliko se naiđe na labelu prverava se da li je ona već u tabeli simola. U objekat klase Simbol upisuje se njena vrednost, tj. vrednost brojača u toku tečuće sekcije. Ispravno je navođenje labele i u istom redu sa drugim kodom i samostalno u redu. Program dozvoljava navođenje dve ili više labela jednu za drugom, ali ne u istom redu. Mora postojati razmak između znaka dvotačke i direktive ili instrukcije koja sledi.

Ukoliko se naiđe na .global asembler sve što se nalazi u istom redu sa ovom direktivom i međusobno je odvojen zarezima tretira se kao deklaracija novog globalnog simbola I upisuje u tabelu simbola, ako u njoj prethodno ne potoji simbol tog imena. Ako postoji takav simbol u tabeli samo se ažurira njegova lokalnost.

Ako se u čitanju naiđe na direktivu skip brojač se povećava za broj bajtova naveden nakon direktive.

Za direktive char, long i word brojač se uvećava za 1, 2 odnosno 4 bajta. Dozvoljeno je navođenje više vrednosti odvojenih zarezima. Zarezi mogu I ne moraju biti bilo kako odvojeni od tih vrednosti.

Kod direktive align traži se prvi broj veći ili jednak brojaču deljiv sa brojem navedenim iza direktive.

Za svaku instrukciju proveravaju se njeni operandi. Ako je kod svih operanada upotrebljeno registarsko direktno adresiranje, brojač se povećava za dva. U svim ostalim slučajevima brojač se povećava za 4.

U drugom prolazu kada se naiđe na direktive char, word ili long čitaju se vrednosti navedene nakon njih. Ukoliko se radi o konkretnom broju u sadržaj odgovarajuće sekcije upisuje se njegova vrenost, raspored bajtova u reči je little-endian. Validno je i iza direktiva word i long navesti naziv neke labela. Ako je labela lokalna u sadržaj odgovarajuće sekcije upisuje se njena vrednost iz tabele simbola, raspored bajtova u reči je little-endian. Pravi se i relokacioni zapis gde ofset predstavlja adresu unutar tekuće sekcije gde se nalazi početak labele, tip relokacije je R\_386\_32, a kao vrednost navodi se redni broj sekcije iz tabele simboola u kojoj se nalazi ova labela. Ako je labela globalna u dodatna dva bajta upisuju se sve nule umesto njene vrednosti, a kao vrednost u relokacionom zapisu se stavlja vrednost te labele iz tabele simbola.

U slučaju direktiva skip i align prostor predviđen za njih na način koji je gore opisan popunjava se nulama.

Instrukcije je validno navoditi samo u text sekciji. Svaka instrukcija mora se nalaziti u zasebnom redu. Najpre se u text sekciju upisuje kod instrukcije. U zavisnosti koja je vrsta operanada provera se da li je oa validna za datu instrukciju. U instrukciji call nije validno neposredno adresiranje. U instrukciji pop nije validno navesti neposredno adresiranje. Adresa destinacije, tj. prvi operand instrukcije ne sme biti neposredno adresiran. Najmanje jedan operand instrukcija koje zahtevaju dva operanda mora biti adresiran registarski direktno.

Ukoliko je za neki način adresiranja navedena labela, njen naziv se traži u tabeli simbola. Ukoliko ona tamo ne postoji ispisuje se greška. Ukoliko je ta labela lokalna u dodatna dva bajta upisuje se njena vrednost, raspored bajtova u reči je little-endian. Pravi se i relokacioni zapis gde ofset predstavlja adresu unutar tekuće sekcije gde se nalazi početak labele, tip relokacije je R\_386\_32, a kao vrednost navodi se redni broj sekcije iz tabele simboola u kojoj se nalazi ova labela. Ako je labela globalna u dodatna dva bajta upisuju se sve nule umesto njene vrednosti, a kao vrednost u relokacionom zapisu se stavlja vrednost te labele iz tabele simbola.

Kod pc relatinog adresiranja je validno iza znaka $ navesti samo ime labele. Nije dozvoljeno navesti broj. U dodatna dva bajta upisuje se vrednost -2 ako je simbol globalan. Ako je simbol lokalan upisuje se njegova vrednost iz tabele simbola smanjena za dva. Tip relokacionog zapisa je R\_386\_PC32, a ostala polja se popunjavaju kao i kod ostalih tipova adresiranja.

Pseudo instrukcija ret se obrađuje kao instrukcija pop r7. Pseudo instrukcija: jmp operand se prevodi kao mov r7, operand, ako je operand adresiran na bilo koji drugi način osim pc relativnim adresiranjem. Ako je operand adresiran pc relativno: jmp $operand se prevodi kao add r7, operand. Relokacioni zapisi se prave kao da su ove napisane u nekom od ovih oblika.

**Uputstvo za pokretanje programa:**

Da bi se pokrenuo program za Linux x86 potrebno je imati instaliran g++. Instalacija se vrši komandama:

sudo add-apt-repository ppa:ubuntu-toolchain-r/test

sudo apt-get update

sudo apt-get remove g++

sudo apt-get install g++-6

Prevođenje programa se vrši komandom:

g++-6 -std=c++11 Main.cpp Asembler.cpp Asembler.h Simbol.h Sifra.h Sifra.cpp RelZapis.h -o program

Pokretanje programa:

./program ulaz.txt izlaz.txt pocetna\_adresa

Gde ulaz.txt predstavlja tekstualni fajl iz koga se čitaju instrukcije. Izlaz.txt je fajl u koji će biti ispisani tabela simbola, relokacioni zapisi i sadržaji sekcija. Pocetna\_adresa je adresa od koje će se u memoriju smeštati sadržaj. Ukoliko izlaz.txt ne postoji, biće automatski kreiran. Ukoliko postoji, svaki sadržaj koji se nalazi u njemu biće izbrisan pre ispisa prgrama.

**Test 1:**

.data

a: .word 5

b: .char 7, 4

c: .align 8

.global b

.global d

d: .skip 3

.text

aladd a, r4

gtmul r3, b

eqcmp r3[c], r0

neshl \*47, r1

alpush 39

gtpop r5[12]

.word 19

gtsub r1, &l

gtcall m

.rodata

.skip 5

m: .word 3

.bss

l: .word 20

.end

**Rezultat 1:**

ime rb sekcija vr. l/g flegovi vel. pocetak

.bss 9 bss 0 l AW 2 1052

.data 1 data 0 l ARW 11 1000

.rodata7 rodata 0 l AR 7 1045

.text 6 text 0 l ARX 34 1011

a 2 data 0 l / -1 /

b 3 data 2 g / -1 /

c 4 data 4 l / -1 /

d 5 data 8 g / -1 /

l 10 bss 0 l / -1 /

m 8 rodata 5 l / -1 /

#data

0500070400000000000000

#text

C20C000089700000136804007A092F00E4002700ABA00C00130085200000AC100500

#rodata

00000000000300

#rel text

ofset tip vr

2 PC\_ABS 1

6 PC\_ABS 3

10 PC\_ABS 1

28 PC\_ABS 9

32 PC\_ABS 7

#rel data

ofset tip vr

#rel rodata

ofset tip vr

**Test 2:**

.data

a:

.char 1, 2, 3

.align 8

.global b

b:

c:

.word 123

.word a

.text

d: almov \*100, r0

eqadd r1, $b

nepop r5[a]

alshr r4, 3

eqdiv r1, &c

gtjmp d

alsub m, r0

.rodata

m: .long 123456

.bss

.skip 5

.end

**Rezultat 2:**

ime rb sekcija vr. l/g flegovi vel. pocetak

.bss 9 bss 0 l AW 5 1044

.data 1 data 0 l ARW 12 1000

.rodata7 rodata 0 l AR 4 1040

.text 5 text 0 l ARX 28 1012

a 2 data 0 l / -1 /

b 3 data 8 g / -1 /

c 4 data 8 l / -1 /

d 6 text 0 l / -1 /

m 8 rodata 0 l / -1 /

#data

01020300000000007B000000

#text

F6086400013FFEFF6BA00000FD8003000D200800B5F00000C6080000

#rodata

40E20100

#rel text

ofset tip vr

6 PC\_REL 3

10 PC\_ABS 1

18 PC\_ABS 1

22 PC\_ABS 5

26 PC\_ABS 7

#rel data

ofset tip vr

10 PC\_ABS 1

#rel rodata

ofset tip vr

**Test 3:**

.data

a: .long 1234567

b: .word a

.global b

.text

gtjmp $t

k:

almul r1, r2

gtiret

t:

alshl r0, 3

eqcmp r3, &b

necall k

eqret

.rodata

.skip 3

m: .char 8

.end

Rezultat:

ime rb sekcija vr. l/g flegovi vel. pocetak

.data 1 data 0 l ARW 6 1000

.rodata7 rodata 0 l AR 4 1028

.text 4 text 0 l ARX 22 1006

a 2 data 0 l / -1 /

b 3 data 4 g / -1 /

k 5 text 4 l / -1 /

m 8 rodata 3 l / -1 /

t 6 text 8 l / -1 /

#data

87D612000000

#text

81E00600C92AB000F9000300116000006C10040029E0

#rodata

00000008

#rel text

ofset tip vr

2 PC\_REL 4

14 PC\_ABS 3

18 PC\_ABS 4

#rel data

ofset tip vr

4 PC\_ABS 1

#rel rodata

ofset tip vr