**OPIS RESENJA**

Asembler se pokrece iz main.cpp fajla gde dobija argumente I otvara input I output fajlove za dalji upis ili ispis. Zatim poziva staticke funkcije prviProlaz() I drugi prolaz() koji predstavljaju dva prolaza kroj koji ulazni fajl prolazi I koji vracaju string koji cemo pri zavrsetku obe funkcije upisati u output.o fajl.

Prvi prolaz parsira ulazni tekst na linije I brize blanko ili tab znakove na pocetku da bi bili spremni za dalje parsiranje. Rekurzivnim metodama dolazimo do tri glavne funkcije: unesiDirektivu(), unesiLabelu() I unesiInstrukciju(). Direktive I Instrukcije prvo parsiramo da bi dobili naziv direktive ili instrukcije I njihove operande. To je reseno pomocu dva enuma koji su lista svih mogucih direktiva I instrukcija.

Od direktiva nalazimo sekcije I unosimo ih u tabelu simbola koja je staticka lista klase Symbol, nakon cega takodje resetujemo locationCounter na nula. Sto se tice ostalih direktiva proveravamo deo uslova (ostatak se proverava u drugom prolazu) I povecavamo locationCounter shodno tipu direktive I broju operanada.

UnesiInstrukciju() nalazi ime I operande instrukcije, nakon cega pozivom funkcija detektujAdresiranje() I adresiranjeBrojBajtova() dobijamo broj bajtova za koji treba da povecamo locationCounter.

Funkcija unesiLabelu() unosi zadatu labelu u tabelu simbola ukoliko je vec nema.

Nakon toga ulazimo u drugi prolaz koji ima staticku listu klase Sekcija koja sadrzi dve liste koje predstavljaju Relokacionu tabelu I Data tabelu. U drugom prolazu prolazimo kroz iste funckije sa indeksom 2.

UnesiInstrukciju2(), unesiDirektivu2() I unesiLabelu2() rade slicnu stvar kao I prethodnici samo sto jos upisuju u dve nove liste podatke I instrukcije prevedene u masinski kod kao I updateovanje tabele simbola.

Na kraju se rezultat oba prolaza upisuje u output.o fajl gde se nalazi tabela simbola, tabela relokacionih zapisa I data tabela svake sekcije sto se dalje prosledjuje linkeru.

**Uputstvo za prevođenje i pokretanje čitavog sistema**

Potrebno je updatovati gcc I g++ da bi se uspesno izvrsilo I to se radi izvrsavanjem sledecih komadni u terminalu operativnog sistema:

sudo add-apt-repository ppa:ubuntu-toolchain-r/test  
sudo apt-get update  
sudo apt-get install gcc-4.9  
sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.9 50  
sudo apt-get install g++-4.9  
sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.9 50

Da bi preveli sve .cpp I .h fajlove moramo napraviti .Makefile koji ce u sebi povezati sve navedene fajlove I pokrenuti g++ sa c++ verzijom 11. Makefile se nalazi u /src folderu.

Treba da se ubaci u folder gde se nalazi svi fajlovi zatim u komandnoj liniji preko komande *cd* doci do zadatog direktorijuma I pokrenuti komandu *make ili make -f Makefile*.

Nakon toga ce se u istom folderu pojaviti Executable file koji trebamo da pokrenemo sa ./exefajl I ukucatu potrebne argumente, u ovom slucaju -o output.o input.s sto ce pri pokretanju I uspesnom izvrsavanju napraviti output.o fajl u kome ce se nalaziti zeljeni rezultat.

**Testovi**

**ULAZ1:**

.global d,m

.text

a:

add r1,r1

addb r1,r2

addw r1,r2

add r1,10

sub r2,r3

mul c,r1

div \*20,r1

cmp r3[10],r3

and r5[e],r1

or r5[e],r1

c:

not r1

push r2

pop r2

shl r4,r5

shr \*21,r1

iret

call c

.data

dataLabela: .char 1,2,3

.skip 10

d:

.long 1500000,11

.align 4

e:

.word 6,7

.rodata

s: .char 4,5

.bss

k: .skip 8

.align 4

.end

**IZLAZ1:**

NAME | TYPE | VALUE | SECTION | SCOPE | INDEX | SIZE | RWX

UND | sekcija | 0 | 0 | l | 0 | 0 | rw

text | sekcija | 0 | 1 | l | 1 | 100 | x

a | simbol | 0 | 1 | l | 2 | 0 |

c | simbol | 69 | 1 | l | 3 | 0 |

data | sekcija | 0 | 4 | l | 4 | 36 | rw

dataLabela | simbol | 0 | 4 | l | 5 | 0 |

d | simbol | 13 | 4 | g | 6 | 0 |

e | simbol | 32 | 4 | l | 7 | 0 |

rodata | sekcija | 0 | 8 | l | 8 | 2 | r

s | simbol | 0 | 8 | l | 9 | 0 |

bss | sekcija | 0 | 10 | l | 10 | 8 | rw

k | simbol | 0 | 10 | l | 11 | 0 |

#.ret.text

OFFSET | TYPE | VALUE

19 | R\_386\_32 | 1

33 | R\_386\_32 | 4

38 | R\_386\_32 | 4

58 | R\_386\_32 | 1

#.ret.data

OFFSET | TYPE | VALUE

#.ret.rodata

OFFSET | TYPE | VALUE

#.text

2c 22 22

28 22 24

2c 22 24

2c 22 00 0a 00

34 24 26

3c a0 45 00 22

44 a0 14 00 22

4c 66 0a 26

5c 8a 20 00 22

64 8a 20 00 22

54 22

8c 24

94 24

7c 28 2a

84 a0 15 00 22

c8

bc a0 45 00

#.data

01 02 03

00 00 00 00 00 00 00 00 00 00

60 e3 16 00 0b 00 00 00

00 00 00 00 00 00 00 00 00 00 00

06 00 07 00

#.rodata

04 05

**ULAZ2:**

.global a, c

.extern b

.text

jeq $a

jeq $e

jeq $b

jeq $d

d: .long d

mov r0, &b

mov c, r0

mov r0, e

.data

.skip 8

e: .long a

.long c

.long bss

a: .long b

.bss

c: .skip 8

.end

**IZLAZ2:**

NAME | TYPE | VALUE | SECTION | SCOPE | INDEX | SIZE | RWX

UND | sekcija | 0 | 0 | l | 0 | 0 | rw

b | simbol | 0 | 0 | g | 1 | 0 |

text | sekcija | 0 | 2 | l | 2 | 41 | x

d | simbol | 16 | 2 | l | 3 | 0 |

data | sekcija | 0 | 4 | l | 4 | 24 | rw

e | simbol | 8 | 4 | l | 5 | 0 |

a | simbol | 20 | 4 | g | 6 | 0 |

bss | sekcija | 0 | 7 | l | 7 | 8 | rw

c | simbol | 0 | 7 | g | 8 | 0 |

#.ret.text

OFFSET | TYPE | VALUE

2 | R\_386\_PC32 | 6

6 | R\_386\_PC32 | 4

10 | R\_386\_PC32 | 1

14 | R\_386\_PC32 | 2

16 | R\_386\_32 | 2

23 | R\_386\_32 | 1

27 | R\_386\_32 | 8

33 | R\_386\_32 | 4

#.ret.data

OFFSET | TYPE | VALUE

8 | R\_386\_32 | 6

12 | R\_386\_32 | 8

16 | R\_386\_32 | 7

20 | R\_386\_32 | 1

#.text

a4 8e fe ff

a4 8e 06 00

a4 8e fe ff

a4 8e 0e 00

10 00 00 00

24 20 00 00 00

24 a0 00 00 20

24 20 a0 08 00

#.data

00 00 00 00 00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

**ULAZ3:**

.global d,m

.text

a:

.data

dataLabela: .char 1,2,3

.skip 10

d:

.long 1500000,11

.align 4

e:

.word 6,7

.rodata

s: .char 4,5

.bss

k: .skip 8

.align 3

.end

**IZLAZ3:**

NAME | TYPE | VALUE | SECTION | SCOPE | INDEX | SIZE | RWX

UND | sekcija | 0 | 0 | l | 0 | 0 | rw

text | sekcija | 0 | 1 | l | 1 | 0 | x

a | simbol | 0 | 1 | l | 2 | 0 |

data | sekcija | 0 | 3 | l | 3 | 36 | rw

dataLabela | simbol | 0 | 3 | l | 4 | 0 |

d | simbol | 13 | 3 | g | 5 | 0 |

e | simbol | 32 | 3 | l | 6 | 0 |

rodata | sekcija | 0 | 7 | l | 7 | 2 | r

s | simbol | 0 | 7 | l | 8 | 0 |

bss | sekcija | 0 | 9 | l | 9 | 8 | rw

k | simbol | 0 | 9 | l | 10 | 0 |

#.ret.text

OFFSET | TYPE | VALUE

#.ret.data

OFFSET | TYPE | VALUE

#.ret.rodata

OFFSET | TYPE | VALUE

#.text

#.data

01 02 03

00 00 00 00 00 00 00 00 00 00

60 e3 16 00 0b 00 00 00

00 00 00 00 00 00 00 00 00 00 00

06 00 07 00

#.rodata

04 05