Univerzitet u Beogradu – Elektrotehnički fakultet



**Projektna dokumetacija**

Sovilj Stefan - 2015/0222

Sistemski Softver 20.8.2018

**Opis problema**

Zadatak koji je zadat kao projektni na predmetu sistemski softver je realizacija dvoprolaznog asemblera za procesor koji je zadat (nesto o tome kasnije). Naime projekat je potrebno izraditi na Linux operativnom sistemu na x86 arhitekturi u programskom jeziku C/C++ ili asembleru. Dvoprolazni asembler treba da ima ulaz u obliku tekstualnog fajla koja je u skladu sa zadatom sintaksom. Izlaz asemblera je predmetni program na skolskoj varijanti ELF formata koji po potrebi moze biti dopunjen.Takodje ulaz asemblera treba da ima i pocetnu adresu od kojih će se smeštati sekcije jedna za drugom.

**Opis procesora**

Procesor za koji treba napisati asembler je 16-bitni procesor sa Von-Neuman arhitekturom. Poseduje 8 16-bitnih opštenamenskih registara r0-r7, i pritom se r7 registar koristi i kao PC a r6 kao SP registar. Takodje postoji i PSW registar. Stek raste ka nižim adresama a SP pokazuje na zauzetu lokaciju na vrhu steka (full-descending).

Sekcije koje postoje su:

- .text - sekcija koja sadrži mašinski kod,

- .data - sekcija koja sadrži inicijalizovane podatke,

-.rodata -sekcija koja sadrži inicijalizovane podatke koje nije dozvoljeno menjati

- .bss - sekcija koja sadrži neinicijalizovane podatke,

Od adresiranja su dostupna:

Registarsko adresiranje, memorijsko, neposredno i memorijsko indirektno. U instrukciji nije dozvoljeno neposrednu vrednost navesti kao destinaciju.

Dostupne instrukcije su: „add“, „sub“, „mul“, „div“, „cmp“, „and“, „or“, „not“, „test“, „push“, „pop“, „call“, „iret“, „mov“, „shl“, „shr“ a tu su i pseudo instrukcije „ret“ i „jmp“. Sve instrukcije su veličine 2 bajta. Ukoliko neki parametar zahteva dodatni podatak on se smešta odmah iza instrukcije u naredna dva bajta. Samo jedan operand u instrukciji može da zahteva dodatne bajtove. Sve instrukcije se uslovno izvrsavaju sa uslovima „eq“, „ne“, „gt“, „al“, gde je uslov „al“ podrazumevani.

Direktive koje je potrebno obraditi su „.align“, „.skip“, „.char“, „.word“, i „.long“.

**Opis resenja**

Projekat je odradjen na C++ jeziku za Linux x86 operativnom sistem. Obrada ulaznog fajla je odradjena onako kako asembler zapravo funkcionise u dva prolaza. U prvom prolazu se parsira linija po linija ulaznog teksta, obavlja se posao prvog prolaza dakle popunjava se tabela simbola i proveravaju sitne sintaksne greške ulaznog koda. Potom se radi drugi prolaz gde se detaljno obradjuju sve linije i instrukcije popunjavaju strukture tabela sekcija i relokacionih tabela i na kraju se sve štampa u izlazni fajl u ELF formatu.

Za prevodjenje na linux platformi je potreban g++. Naime pošto na školskim virtuelnim mašinama C++ verzija je previše stara pa je potrebno prvo instalirati noviju verziju g++-a a potom koristiti noviju verziju C++11. Koristimo sledeće komande da bi ovo postigli:

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

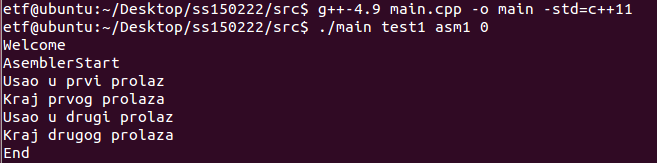
**$ sudo apt-get update**

**$ sudo apt-get install g++-4.9**

Kada smo instalirali g++ verziju 4.9 za prevodjenje programa koristimo sledeću komandu (prvo se pozicioniramo na mesto src fajlova):

**$ g++-4.9 main.cpp -o main -std=c++11**

Na školskoj virtuelnoj mašini ovo treba da izgleda ovako nekako:



Prevodimo sa g++-4.9 formiramo izvršni fajl main i nakon toga pozivamo program sa ulaznim test1 fajlom i želimo izlaz u obliku asm1 fajla.

Na masini koja već poseduje noviju verziju g++ program pozivamo sledećim naredbama (na mestu gde se nalaze fajlovi) :

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

Ova naredba kreira izvrsni fajl „main“ od ulaznog fajla „main.cpp“.

-g je flag za debug, a -std=c++11 govori vise o verziji kompajlera.

Potom pokrećemo naš program sledećom linijom:

**$ ./main <inputFile> <outputFile> <startAdress>**

Npr : $ ./main in.s out.o 0

-inputFile - je tekstualni fajl u kome se nalazi sam kod

-outputFile - je objektni fajl koje predstavlja rezultat asembliranja

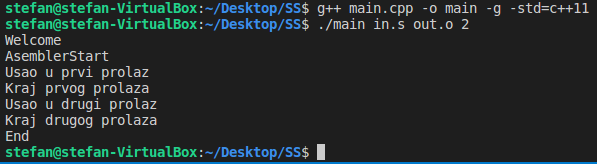
-startAdress - je broj odnostno adresa sa koje treba da počne smeštanje sekcija

Npr:



Kada se program izvrši ukoliko nema nijedne greške u sintaksi ulaznog fajla rezultat je popunjen izlazni fajl.

Primer pokretanja i uspešnog izvršavanja programa:



Izlazni fajl moze da sadrži više tabela od kojih je na početku tabela „InfoSekcija“ u kojoj se nalaze podaci o sekcijama, veličina sekcije i početna adresa sekcija. Ovaj format je ispisan u dekadnom obliku da bi bilo lakše uvideti veličine i početne adrese.

Sledeca tabela je tabela simbola u kojoj se nalazi lista svih simbola njihovih lokacija, pomeraja, opsega i rednih brojeva. Potom slede tabele samih sekcija „.text“ i „rel.text“ ,“data“ i „rel.data“ i na kraju „rodata“ sekcija.

**Testovi**

**Test 1:**

Mali test da se proveri rad sekcija za podatke i direktive. Prilikom poziva testova svuda se koristene iste, gore navedene, komande.

Ulaz:

.global a

.data

.word 5

.word &a

.long 4

c:

.long d

.long &d

.long -14

.bss

.skip 4

d:

.align 10

.rodata

.char 55

.end

Izlaz:

#INFO SEKCIJA:

Start address: 2

Ime: Size: Start:

.data 20 2

.bss 10 22

.rodata 1 32

#TABELA SIMBOLA

Labela: Sekcija: Offset Scope Redni\_broj

UND 00 L 0

.data .data 00 L 1

c .data 08 L 2

.bss .bss 00 L 3

d .bss 04 L 4

.rodata .rodata 00 L 5

a UND 00 G 6

#DATA SEKCIJA:

0500 0000 04000000 04000000 04000000 F2FFFFFF

#DATA\_REL SEKCIJA:

Offset: Tip: Vr:

00000002 R\_386\_16 6

00000008 R\_386\_16 3

0000000C R\_386\_16 3

#RODATA SEKCIJA:

37

**Test 2:**

U ovome testu sada aktiviramo i text sekciju i koristimo neke osnovne instrukcije

Ulaz:

.global a,b

.global putchar

.text

mov r1,0

mov r2,20

addgt r2,r1

jmpeq exit

push r1

push r2

call putchar

mov r3,&array

shr r3,1

addne r3,r5[a]

moval r3[0],r2

exit: pop r2

pop r1

iret

.data

value: .long 387

array: .word 1111

.bss

a: .skip 8

.end

Izlaz:

#INFO SEKCIJA:

Start address: 2

Ime: Size: Start:

.text 44 2

.data 6 46

.bss 8 52

#TABELA SIMBOLA

Labela: Sekcija: Offset Scope Redni\_broj

UND 00 L 0

.text .text 00 L 1

exit .text 26 L 2

.data .data 00 L 3

value .data 00 L 4

array .data 04 L 5

.bss .bss 00 L 6

a .bss 00 G 7

b UND 00 G 8

putchar UND 00 G 9

#TEXT SEKCIJA:

DD270000 DD471400 0949 D1F71800 9D3F 9D5F BEFF0000 DD670400 FD670100 057D0000 DF6A0000 AD5F AD3F CFFF

#TEXT\_REL SEKCIJA:

Offset: Tip: Vr:

0000000C R\_386\_PC16 1

00000014 R\_386\_PC16 9

00000018 R\_386\_16 3

00000020 R\_386\_16 7

#DATA SEKCIJA:

83010000 5704

**Test 3:**

Nešto duži test u kome koristim najrazličitije moguće komande na najrazličitijim mestima u cilju stersiranja programa.

Ulaz:

.global a,b,c

.global printf

.text

add r1,r1[4]

jmpgt next

b: testeq r1,c

noteq r1,&b

sub r1,1

shl r2,2

popgt \*20

moval r1,&b

next: push r5[20]

jmpgt $d

jmpne $a

add r1,r4[a]

push 1

push &a

call printf

mul r1,r4

.data

.long 10

a: .word b

info: .char 1

.align 4

.word &a

.word &b

.word &c

.long 1234

.long 5678

.bss

c: .skip 10

.rodata

d: .long 4

.char 55

.end

Izlaz:

#INFO SEKCIJA:

Start address: 2

Ime: Size: Start:

.text 62 2

.data 22 64

.bss 10 86

.rodata 5 96

#TABELA SIMBOLA

Labela: Sekcija: Offset Scope Redni\_broj

UND 00 L 0

.text .text 00 L 1

b .text 08 G 2

next .text 20 L 3

.data .data 00 L 4

a .data 04 G 5

info .data 06 L 6

.bss .bss 00 L 7

c .bss 00 G 8

.rodata .rodata 00 L 9

d .rodata 00 L A

printf UND 00 G B

#TEXT SEKCIJA:

0D390400 D9F71800 81370000 71270800 1D270100 ED470200 AAFF1400 DD270800 9FBF1400 09F72800 05F7FEFF 0D3C0400 9CFF0100 9CFF0400 BEFF0000 2D2C

#TEXT\_REL SEKCIJA:

Offset: Tip: Vr:

00000006 R\_386\_PC16 1

0000000A R\_386\_16 8

0000000E R\_386\_16 2

0000001E R\_386\_16 2

00000026 R\_386\_PC16 9

0000002A R\_386\_PC16 5

0000002E R\_386\_16 5

00000036 R\_386\_16 5

0000003A R\_386\_PC16 B

#DATA SEKCIJA:

0A000000 0800 01 0400 0800 0000 D2040000 2E160000

#DATA\_REL SEKCIJA:

Offset: Tip: Vr:

00000004 R\_386\_16 2

00000008 R\_386\_16 5

0000000A R\_386\_16 2

0000000C R\_386\_16 8

#RODATA SEKCIJA:

04000000 37

**Test 4:**

Test u kome su i za direktive i za sve tipove instrukcija izlistane sve varijante adresiranja radi provere njihovog pojedinačnog funkcionisanja.

Ulaz:

.global x,b,c

.global printf

.data

niz:

.word &cont

.word cont

.word -20

.long -2

.long cont

.long &cont

.char 1

.char 255

.skip 8

.align 20

.rodata

.char 4

.word -3

.long -19

.align 10

.bss

.skip 8

.align 10

.text

a:

pop a

pop \*20

pop r0

pop r0[32]

pop r0[x]

add r0,20

add r0,&a

add r0,a

add r0,\*20

add r0,r1

add r0,r1[32]

add r0,r1[a]

push 20

push &a

push a

push \*20

push r3

push r4[-4]

push r4[a]

jmp x

jmp &x

jmp r2

jmp r3[32]

jmp 20

jmp \*20

jmp $x

cont:

add r0,r0

ret

.end

Izlaz:

#INFO SEKCIJA:

Start address: 2

Ime: Size: Start:

.data 40 2

.rodata 10 42

.bss 10 52

.text 102 62

#TABELA SIMBOLA

Labela: Sekcija: Offset Scope Redni\_broj

UND 00 L 0

.data .data 00 L 1

niz .data 00 L 2

.rodata .rodata 00 L 3

.bss .bss 00 L 4

.text .text 00 L 5

a .text 00 L 6

cont .text 62 L 7

x UND 00 G 8

b UND 00 G 9

c UND 00 G A

printf UND 00 G B

#TEXT SEKCIJA:

AEFF0000 AEFF1400 AD1F AF1F2000 AF1F0000 0D071400 0D070000 0D170000 0D171400 0D09 0D192000 0D190000 9CFF1400 9CFF0000 9EFF0000 9EFF1400 9D7F 9F9FFCFF 9F9F0000 DDF7FEFF DDE7FEFF DDEA DDFB2000 DDE71400 DDF71400 0DF7FEFF 0D08 ADFF

#TEXT\_REL SEKCIJA:

Offset: Tip: Vr:

00000002 R\_386\_16 5

00000010 R\_386\_16 8

00000018 R\_386\_16 5

0000001C R\_386\_16 5

0000002A R\_386\_16 5

00000032 R\_386\_16 5

00000036 R\_386\_16 5

00000044 R\_386\_16 5

00000048 R\_386\_PC16 8

0000004C R\_386\_PC16 8

00000060 R\_386\_PC16 8

#DATA SEKCIJA:

6200 6200 ECFF FEFFFFFF 62000000 62000000 01 FF 0000000000000000

#DATA\_REL SEKCIJA:

Offset: Tip: Vr:

00000000 R\_386\_16 5

00000002 R\_386\_16 5

0000000A R\_386\_16 5

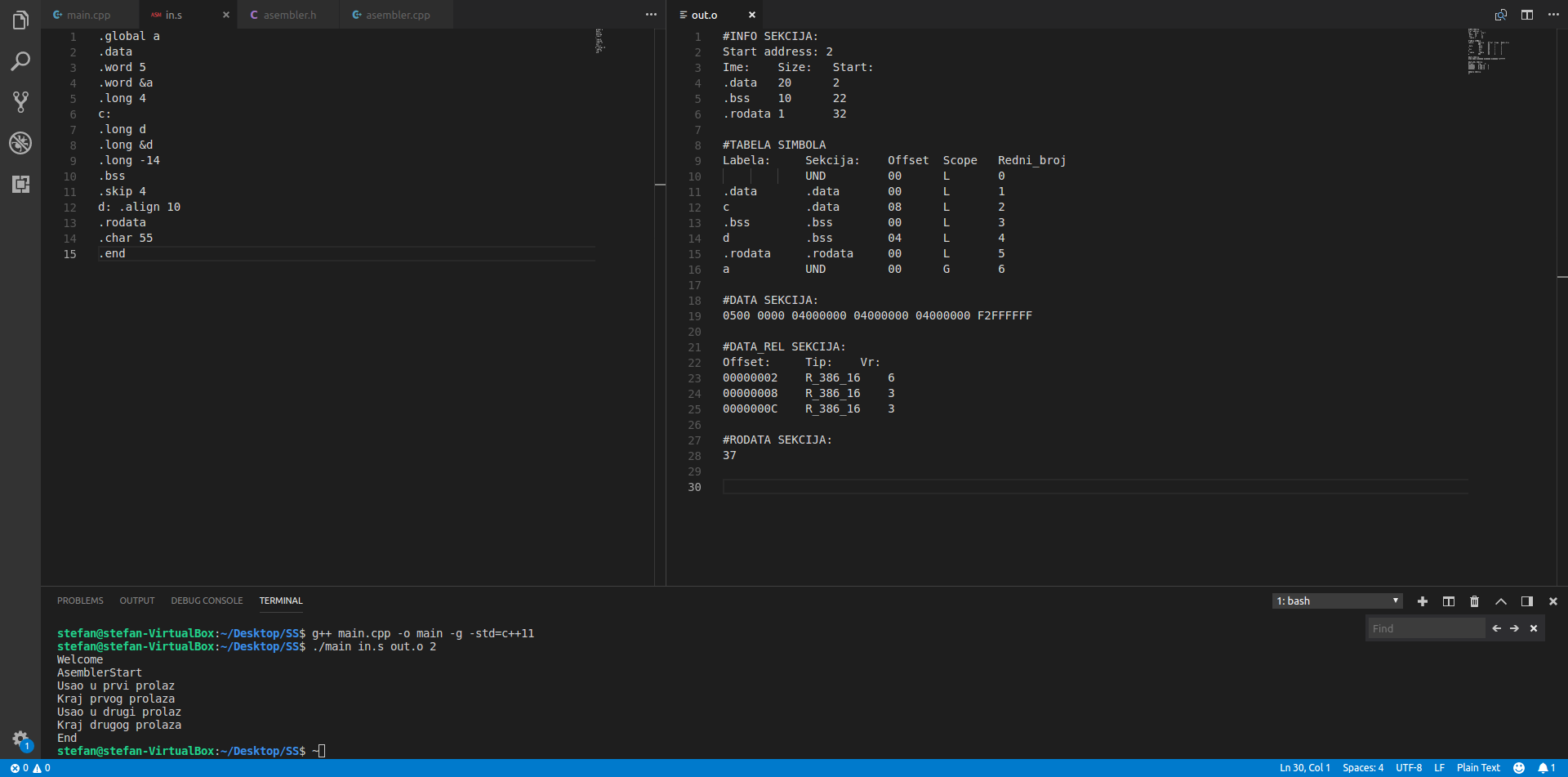
0000000E R\_386\_16 5

#RODATA SEKCIJA:

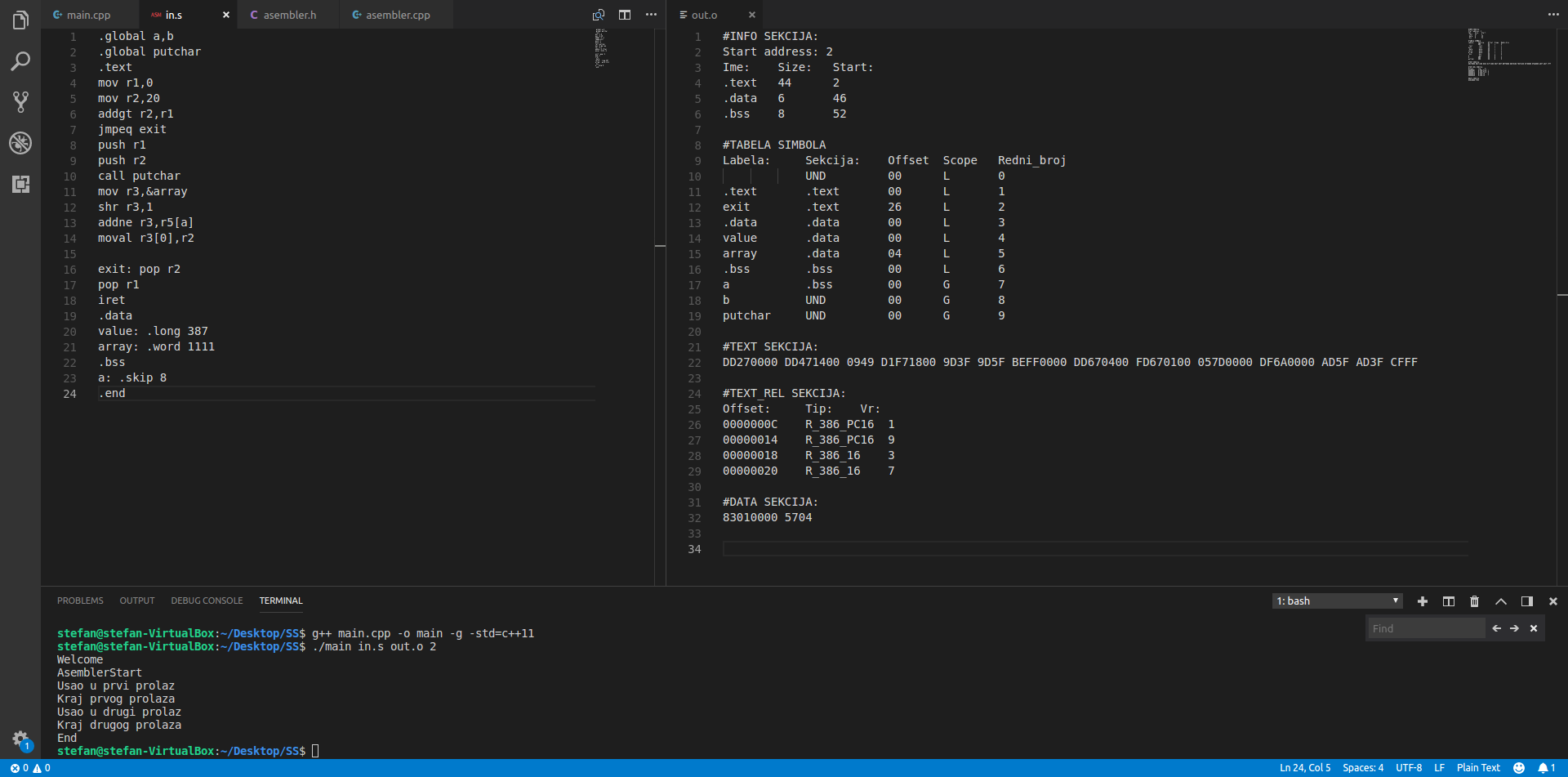
04 FDFF EDFFFFFF

Testovi komplet sa ulaznim fajlom, izlaznim fajlom i terminalom za pokretanje se takodje mogu pogledati i na **slikama** koje se nalaze u dokumentaciji a „preview“ je dat i ovde:

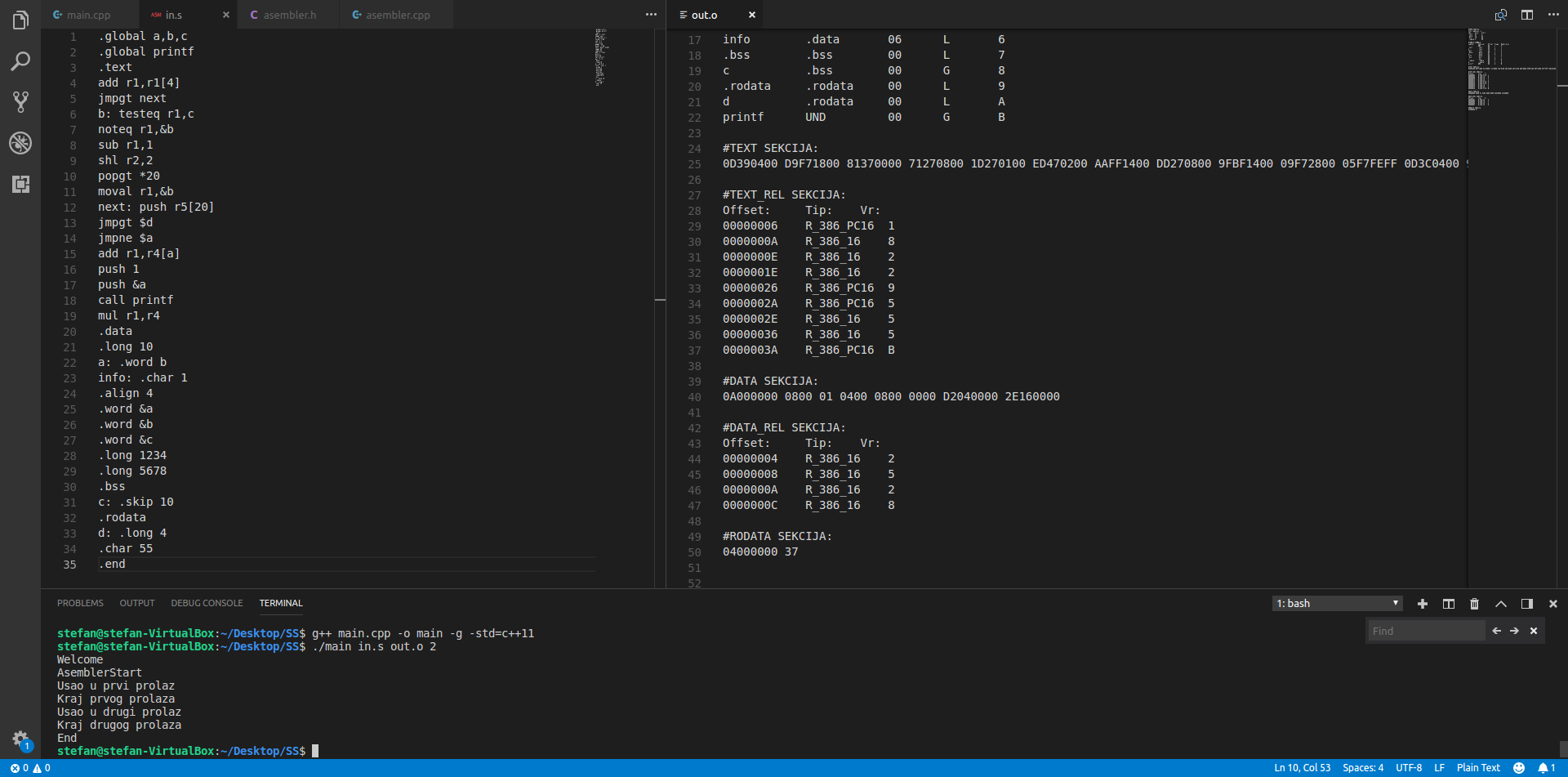
Test1:

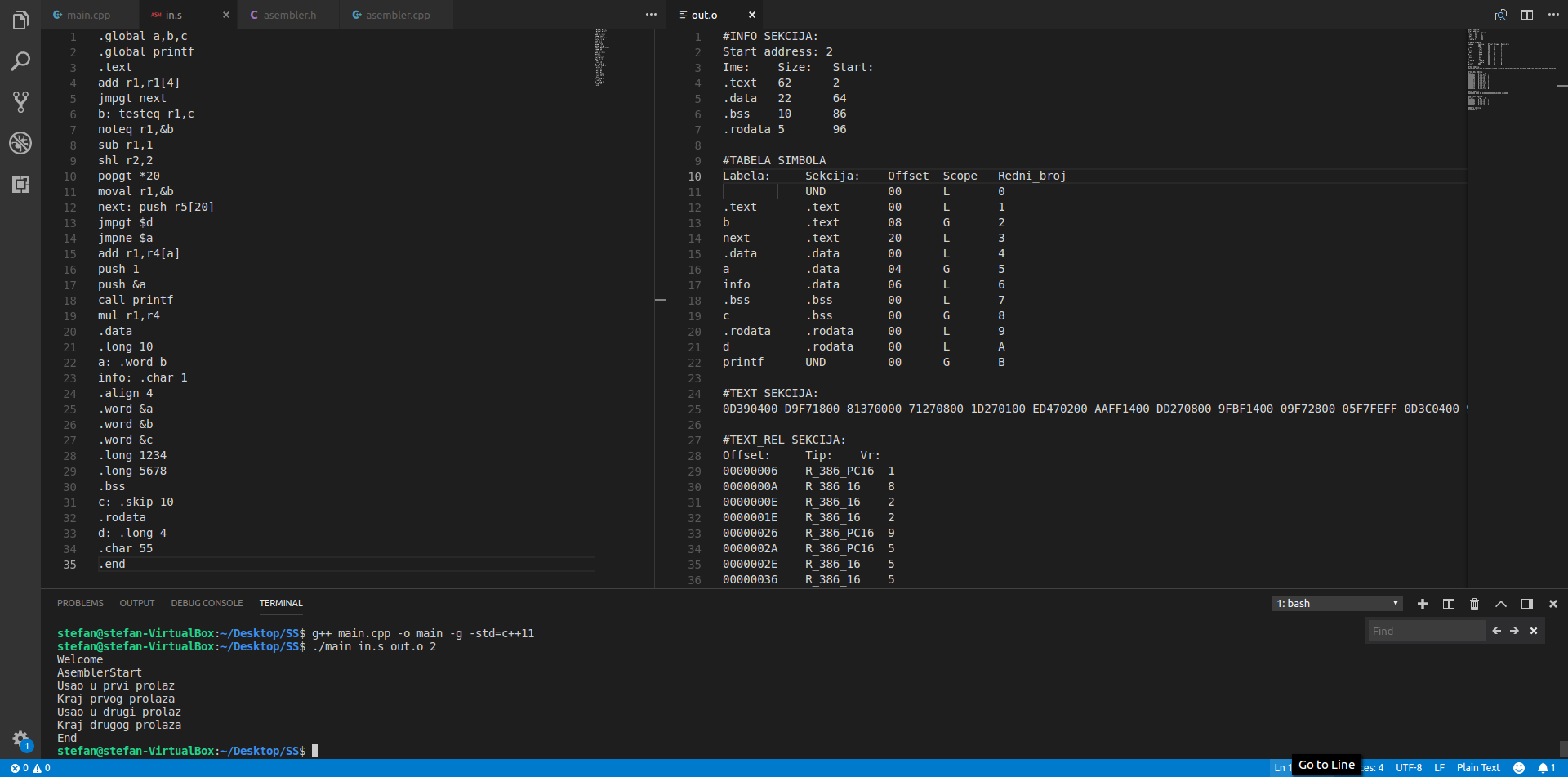


Test2:



Test3:





Test4:

