**Sistemski softver 13E113SS**

*Domaći zadatak - jun 2019*

*Autor: Nevena Cvetković*

*Broj indeksa: 0266/2016*

*Datum: 15.06.2019*

**Opis problema**

Prvi domaći zadatak iz predmeta Sistemski softver je implementacija dvoprolaznog aseblera na Linux operativnom sistemu. Kao ulaz asemblera se prosledjuje tekstualni fajl u kom je napisan asemblerski program za dati procesor. Izlaz asemblera predstavlja tekstualni fajl koji predstavlja objektni fajl koji odgovara datom programu. Izlazni fajl sadrži tabelu simbola, mašinski kod i realokacione zapise koje linker koristi pri povezivanju više takvih asemblerskih programa.

**Opis rešenja**

Rešenje problema je program napisan na jeziku C++. Program kao parametre prima nazive ulaznog i izlaznog tekstualnog fajla. Ulazni i izlazni fajl imaju ekstenziju .txt.

U prvom prolazu asembler nailazi na definicije simbola koje dodaje u tabelu simbola uz postavljanje odgovarajućih atributa. Tabela simbola je posebna struktura cija se definicija nalazi u fajlovima TableOfSymbols.h i TableOfSymbols.cpp. Asembler u prvom prolazu prati dužine svih instrukcija, uz uvećavanje locationCountera kako bi na kraju prvog prolaza bile poznate veličine svih sekcija i vrednosti lokalnih i globalnih simbola. Asembler takodje proverava i da li su adresiranja koja se koriste za operande dozvoljena i da li je veličina operanada odgovarajuća. Pri nailasku na direktive asembler, kod onih kojih je to moguće, proverava validnost njihovih parametara.

Po završetku čitanja fajla asembler na kraju prvog prolaza proverava da li su svi simboli koji se definišu pomoću .equ drektive poznati, ako nisu prijavljuje grešku. Takodje proverava i da li su sve promenljive koje su deklarisane kao gobalne definisane, u suprotnom prijavljuje grešku.

Po završetku svih provera asembler počinje sa drugim prolazom. U drugom prolazu se generišu mašinski kodovi instrukcija i realokacioni zapisi. Mašinski kodovi se generišu prema uputstvu definisanom u prilogu projekta. Realokacioni zapisi se generišu za sve instrukcije i direktive koje referišu promenljive iz tabele simbola jer posle povezivanja i punjenja sekcije neće imati početnu adresu 0 i vrednosti simbola iz tabele simbola neće bitni tačne. Realokacioni zapisi se generišu samo za .text i .data sekcije. Sve potrebne strukture za realokaciju se nalaze u okviru fajlova RealocationTable.h, RealocationTable.cpp, RelRow.h i RelRow.cpp.

**Uputstvo za pisanje programa**

U nastavku su opisana pravila koja važe za pisanje programa na ovom asembleru:

* Komentari pišu u zasebnoj liniji posle karaktera #
* Linije se parsiraju tako da postojanje blanko karaktera ne utiče na značenje sem u slučaju instrukcije sa dva operanda i u slučaju direktiva .global, .extern, .equ
* U slučaju instrukcije sa dva operanda, operandi se moraju navesti isključivo u obiku: operand1, operand2, sa tačno jednim razmakom i zarezom
* U slučaju .global i .extern direktiva, promenljive se moraju navesti isključivo u obliku: var1, var2, sa tačno jednim razmakom i zarezom. Moguće je navesti proizvoljno mnogo promenljivih
* U slučaju .equ direktive promenljiva i izraz se navode u obliku: prom, izraz, sa tačno jednim razmakom i zarezom. Izraz može imati maksimalno tri operanda i dva operatora koji mogu biti isključivo + i -
* Kod direktiva .byte i .word nije dozvoljeno za inicijalizaciju pisati izraz, već isključivo konstantnu vrednost ili vrednost promenljive
* Nije moguće kao destinacioni operand instrukcija sa dva operanda koristiti neposredno adresiranje jer ni greška neće biti precizno detektovana
* Definicija simbola se mora završiti dvotačkom. Dozvoljeno je korišćenje samo malih slova
* Ne postoje ključne reči pc, sp. Umesto njih se koriste registri r7, r6 respektivno.
* Kod registarskih adresiranja, dodavanje karaktera b i w nije podržano

**Uputstvo za prevodjenje programa**

Izvorni kod projekta se prevodi pomoću g++ prevodioca. Neophodno je instalirati verziju 4.9 (i više) kako bi neke biblioteke C++11 bile dostupne za korišćenje. Neophodno je instalirati i 4.9 verziju gcc. Instalacija se obavlja pomoću sledećih komandi:

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

*sudo apt-get update*

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

Komanda za prevodjenje izvornog koda asemblera je:

*g++-4.9 -std=c++11 Main.cpp TableOfSymbols.cpp Symbol.cpp RelRow.cpp RealocationTable.cpp -o projekat*

-std=c++11 je nepohodno navesti tokom prevođenja programa da bi prevodilac znao koja se verzija koristi pa time znao i koje bibilioteke da uvrsti.

Treba se pozicionirati u folder u kom se nalazi izvorni kod asemblera.

**Uputstvo za pokretanje programa**

Nakon uspešnog prevodjenja izvršni fajl će se pojaviti u tekućem direktorijumu. Njegovo pogretanje je moguće pomoću sledeće komande:

*projekat input.txt output.txt*

gde su input.txt i output.txt imena ulaznog i izlaznog fajla.

Kada se asembliranje završi u konzoli će biti ispisane tabela simbola i realokacioni zapisi za tekst i data sekcije. Ako dodje do greške pri izvršavanju biće ispisana poruka o grešci.

**Testovi**

Id Name Section Value Is local Size RWXP

0 und UND 0 local 0 0

6 data DATA UND local 8 WR

11 text TEXT UND local 44 RE

12 bss BSS UND local 0 WR

1 a DATA 0 global -1 0

2 b DATA 2 global -1 0

3 d UND UND global -1 0

4 e UND UND global -1 0

5 f UND UND global -1 0

7 poyy DATA 0 local -1 0

8 nova DATA 2 local -1 0

9 g DATA 3 local -1 0

10 cao DATA 16 local -1 0

13 s BSS 0 local -1 0

#text

#offset #type vr[.text]

0000 R\_386\_32 1

0000 R\_386\_32 2

001e R\_386\_32 6

0022 R\_386\_PC32 6

#data

#offset #type vr[.data]

0000 R\_386\_32 3

0000 R\_386\_32 2

#.text

20 3c 00 00 34 12 20 50 00 00 50 02 00 20 12 14 20 14 00 ff ff 98 12 b0 50 64 00 a0 94 a8 36 64 b8 50 03 00 98 40 00 00 c0

#.data

00 00 00 02 00 00 00 00 00

*Output1,txt*

1. **Test** skokovi

.global a, b

.extern d, e, f

.data

a:.skip 2

.equ poyy, a

.equ nova, b-cao+a

b:.byte d

g: .word b

.equ cao, 10-6+12

.align 4

.text

mov r6[0], 4660

mov a, b

mov r1, r2

mov r2, -1

jmp r1

jgt \*100

jeq [r2]

jne r3[100]

call g

jmp $nova

ret

.bss

s: .long 3

.end

*ulaz1.txt*

1. **Test** aritmetičke i pomeračke instrukcije

Id Name Section Value Is local Size RWXP

0 und UND 0 local 0 0

5 text TEXT UND local 59 RE

7 data DATA UND local 19 WR

1 a DATA 8 global -1 0

2 b DATA 18 global -1 0

3 c DATA 10 global -1 0

4 labela UND UND global -1 0

6 lab TEXT 38 local -1 0

#text

#offset #type vr[.text]

0000 R\_386\_32 1

0000 R\_386\_32 3

0000 R\_386\_32 4

0000 R\_386\_32 2

0000 R\_386\_32 3

002a R\_386\_32 5

0000 R\_386\_32 1

#.text

20 12 00 05 00 20 14 50 08 00 28 12 14 50 94 30 16 50 90 01 38 34 00 46 dc 05 60 46 0a 00 50 00 00 48 50 12 00 50 0a 00 10 32 00 14 70 50 26 00 50 08 00

#.data

00 00 00 00 00 00 00 00 c8 00 00 00 00 00 00 00 00 00 05

.global a, b, c

.extern labela

.text

mov r1, 5

mov r2, &a

add r1, r2

not [r2]

sub r3, \*400

mul r2[0], r3[1500]

or r3[c], labela

lab:

cmp b, c

xchg r1[0], r2

test &lab, &a

.data

.align 8

a: .word 200

c: .skip 8

b: .byte 5

.end

1. **Test** bezadresne i instrukcije sa jednim operandom

Id Name Section Value Is local Size RWXP

0 und UND 0 local 0 0

4 bss BSS UND local 10 WR

7 text TEXT UND local 47 RE

9 data DATA UND local 7 WR

1 a DATA 0 global -1 0

2 b DATA 2 global -1 0

3 eksterna UND UND global -1 0

5 c BSS 0 local -1 0

6 d BSS 8 local -1 0

8 labela TEXT 19 local -1 0

#text

#offset #type vr[.text]

0000 R\_386\_32 1

0000 R\_386\_32 2

0000 R\_386\_32 3

0013 R\_386\_32 7

0000 R\_386\_32 2

001b R\_386\_32 4

#.text

88 50 00 00 88 50 02 00 88 50 00 00 88 12 88 94 88 3c 04 90 12 90 94 90 4c 13 00 90 50 02 00 90 50 00 00 18 12 18 00 05 00 18 50 05 00 c0 c0 08

#.data

05 00 00 00 00 00 00

.global a, b

.extern eksterna

.bss

c: .skip 8

d: .word 15

.text

push a

push &b

push &eksterna

push r1

push [r5]

push r6[4]

labela:

pop r1

pop [r2]

pop r6[labela]

pop b

pop c

int r1

int 5

int \*5

ret

iret

halt

.data

a: .word 5

b: .skip 5

.aliign 4

.end

1. **Test** greške

**Error! Multiple text sections!**

**Error! Invalid number of operands!**

**Error! Instructions can be written only in .text section!**

.data

c: .byte 5

mov c, r1

.end

*instrOutOfText.txt*

.text

mov 5, r1

add &a, r2

a:

xchg [r3], &a

.end

#to test every error individually place # at the begining of a tested instruction

*invalidOperands.txt*

.global a

.text

mov r1, a

pop r2

.text

.end

*errorMultSections.txt*