Elektrotehnički fakultet Univerziteta u Beogradu

Katedra za Računarsku tehniku i informatiku

**Dvoprolazni asembler**

Opis problema i rešenje zadatka

Student: professor: Dragan Bojić

Ilija Stevanović 2011/0350 asistent: Saša Stojanović

JUN, 2017

Opis problema:

Potrebno je implementirati dvoprolazni asembler. Ulaz je tekstualni fajl u skladu sa sintaksom definisanom postavkom domaćeg zadatka. Izlaz je tekstualni fajl, koji predstavlja predmetni program koji je takođe definisan postavkom domaćeg zadatka, sličan ELF formatu. Projekat se radi pod operativnim sistemom Linux (Ubuntu 12.04 32bit). Programski jezici koji se mogu koristiti su C, C++, asembler, ili njihova kombinacija.

Rešenje problema:

U prilozenom rešenju realizovane su tri strukture podataka:

-Tabela simbola

-Tabela sekcija

-Tabela relokacija

Ove strukture se koriste u klasi “Assembler.cpp” koja simulira ponašanje dvoprolaznog asemblera, čije su karakteristike opisane u prilogu. “Prolaz” je implementiran kao funkcija koja kao argument prima broj prolaza, 1 ili 2.

Prvi argument koji se zadaje prilikom pokretanja programa predstavlja ulaznu tekstualnu datoteku, koja predstavlja asemblerski program koji je potrebno obraditi.

Cilj prvog prolaza je (delimicno) formiranje tabele simbola, utvrdjivanje pocetnih adresa sekcija i njihovih velicina, odnosno prikupljanje osnovnih informacija koje će se koristiti u ponovnom, drugom prolazu asemblera.

Cilj drugog prolaza je kompletiranje finalnog izgleda tabele simbola, određivanja sadržaja sekcija koje imaju sadržaj, kao i spremanje informacija za linker u vidu formiranja tabele relokacija.

Klase koje su navedene na početku izlaganja rešenja problema sadrže funkcije koje vrše upis u tekstualnu datoteku, koja je navedena kao drugi argument prilikom pokretanja programa. Ispis tabele simbola, sekcija i relokacija je u skladu sa postavkom domaćeg zadatka. Pored priloženog rešenja, dostupna 3 test fajla, kao i njihovi odgovarajući izlazi.

**Komande za pokretanje:**

Iz terminala, potrebno je prebaciti se u direktorijum gde se nalaze fajlovi. U istom direktorijumu treba da budu cpp i h fajlovi kao i test primeri. Izlaz programa je output.bin fajl. Po potrebi, potrebno je unaprediti alate koji se koriste sledećim komandama:

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

sudo apt-get update

sudo apt-get install gcc-5 g++-5

sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 60 --slave /usr/bin/g++ g++ /usr/bin/g++-5

*kompajliranje:*

g++ symboltable.cpp symbol.cpp sectiontable.cpp section.cpp relocationtable.cpp relocation.cpp assembler.cpp main.cpp -std=c++0x -g -o asm.out –fpermissive

*pokretanje:*

./asm.out ulaznadatoteka.txt izlaznadatoteka.txt

**TEST PRIMER 1:**

.data

DW a, b, c

a DEF 500

b DEF 200

c DEF 0x20

.extern prom

.text.1

PUSH R1

x: PUSH R2

LOAD R1, R2

y:

LOAD R3, [R4]

LOADSW R11, x

JLEZ R14, $x

LOAD R3, #a

STORESW PC, nepostojecaprom

ORG 0b1000+333-(20-(5+3))

.text

DB 5-3 DUP 0xF

JLEZ R1, y

.text.100

LOADSW R4, #0xF

CALL $prom

POP R1

POP R2

SUB R14, R15, R16

.end

**OUTPUT PRIMER 1:**

#TabelaSimbola

#symbol type #ID #name #section #start adr #size #flags

SEG/SYM 0 UND 0 0 0 l

SEG 1 .data 1 0 12 WRITE, ALLOC

SEG 2 .text.1 2 0 52 EXECINSTR, ALLOC

SEG 3 .text 3 329 12 EXECINSTR, ALLOC

SEG 4 .text.100 4 0 28 EXECINSTR, ALLOC

SYM 5 a 1 500 0 l

SYM 6 b 1 200 0 l

SYM 7 c 1 32 0 l

SYM 8 prom 0 0 0 g

SYM 9 x 2 4 0 l

SYM 10 y 2 12 0 l

SYM 11 nepostojecaprom 0 0 0 g

#SadrzajSekcija

<.data>

f4 01 00 00

c8 00 00 00

20 00 00 00

<.text.1>

20 01 00 00

20 02 00 00

10 01 10 00

10 43 20 00

10 eb 00 28

04 00 00 00

09 ee 88 00

eb ff ff ff

10 83 00 00

f4 01 00 00

11 e1 00 28

00 00 00 00

<.text>

0f 00 0f 00

09 e1 00 00

0c 00 00 00

<.text.100>

10 84 00 28

0f 00 00 00

03 f1 00 00

fc ff ff ff

21 01 00 00

21 02 00 00

31 0e 7c 00

#rel.data

#offset #type #symbol ID

0x0 A 1

0x4 A 1

0x8 A 1

#rel.text.1

#offset #type #symbol ID

0x14 A 2

0x20 A 1

0x28 A b

#rel.text

#offset #type #symbol ID

0x151 A 2

#rel.text.100

#offset #type #symbol ID

0x8 R 8

**TEST PRIMER 2:**

.data

c DEF 300

DB 1, 2, 3

DW 5+((50+5)-(5-33))

DB a, 0x20, 0b0

.text\_1

a DEF 13

LOADSB R1, R2

LOAD R5, $20

LOAD PC, $a

CALL nepostojecisimbol ; iako simbol ne postoji, bice ubacen u tabelu u drugom prolazu

.text\_2

CALL a

CALL $a

ORG 0x20-0b101+(33-22-(21+3))

.bss

DW 5 DUP ?

.end

**OUTPUT PRIMER 2:**

#TabelaSimbola

#symbol type #ID #name #section #start adr #size #flags

SEG/SYM 0 UND 0 0 0 l

SEG 1 .data 1 0 16 WRITE, ALLOC

SEG 2 .text\_1 2 0 28 EXECINSTR, ALLOC

SEG 3 .text\_2 3 0 16 EXECINSTR, ALLOC

SEG 4 .bss 4 14 20 WRITE, ALLOC

SYM 5 c 1 300 0 l

SYM 6 a 2 13 0 l

SYM 7 nepostojecisimbol 0 0 0 g

#SadrzajSekcija

<.data>

01 00 02 00

03 00 58 00

00 00 0d 00

20 00 00 00

<.text\_1>

10 01 10 18

10 e5 88 00

10 00 00 00

10 e1 88 00

01 00 00 00

03 e0 00 00

fc ff ff ff

<.text\_2>

03 e0 00 00

09 00 00 00

03 f1 00 00

09 00 00 00

<.bss>

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

00 00 00 00

#rel.data

#offset #type #symbol ID

0xa A 2

#rel.text\_1

#offset #type #symbol ID

0x14 A 7

#rel.text\_2

#offset #type #symbol ID

0x0 A 2

0x8 R 2

#rel.bss

#offset #type #symbol ID

**TEST PRIMER 3:**

ORG 0x20

.text

JNZ R5, R4

PUSH PC

POP PC

bp1:PUSH SP

POP SP

ADD R1, R2, R3

JGEZ R7, $bp1

SUB R4, R5, R6

MUL R7, R8, R9

DIV R10, R11, R12

MOD R13, R14, R15

bp2:AND R15, R14, R13

OR R12, R11, R10

XOR R9, R8, R7

NOT R6, R5

ASL R4, R3, R2

.end

**OUTPUT PRIMER 3:**

#TabelaSimbola

#symbol type #ID #name #section #start adr #size #flags

SEG/SYM 0 UND 0 0 0 l

SEG 1 .text 1 32 60 EXECINSTR, ALLOC

SYM 2 bp1 1 44 0 l

SYM 3 bp2 1 76 0 l

#SadrzajSekcija

<.text>

05 05 20 00

20 11 00 00

21 11 00 00

21 10 00 00

30 01 10 c0

07 e7 88 00

f7 ff ff ff

31 04 29 80

32 07 42 40

33 0a 53 00

34 0d 73 c0

36 0c 5a 80

37 09 41 c0

38 06 28 00

39 04 18 80

#rel.text

#offset #type #symbol ID