Elektrotehnički fakultet

Univerziteta u Beogradu

Dokumentacija za prvi deo projekta iz predmeta Sistemski softver

Mentor: Student:

Marko Mićović Sara Lazić,

broj indeksa: 2016/0054, smer: RTI

Beograd, avgust 2020.

U nastavku je opisana implementacija prvog dela zadatka iz predmeta Sistemski softver, odnosno implementacija jednoprolaznog asemblera. Opisane su korišćene klase kao i njihove metode.

Prilikom implementacije korišćene su klase: **Convert, Instructions, Symbol\_table, Relocation\_list i One\_pass**.

Klasa **Convert** sadrži sledeće metode:

* **static string BinToHex(string bin);** - prima argument koji predstavlja binarni broj i kao rezultat vraća njegov heksadecimalni ekvivalent
* **static string DecToHex(int dec); -** prima celobrojni argument i vraća njegov heksadecimalni ekvivalent
* **static string DecToBin1Digit(int dec); -** prima celobrojni argument (manji od 16) i vraća njegov binarni ekvivalent (četiri bita)
* **static char\* StringToCharArr(string str, char from, char without, char to);** - prima kao argumente string, karakter od koga treba da kopiramo string, karakter koji treba da izostavimo, i karakter do koga kopiramo, respektivno. Ako je za neki od poslednja tri argumenta uneta vrednost prazan karakter, smatra se da taj kriterijum ne postoji (npr. ako se za from unese prazan karakter, string se kopira od pocetka).

Klasa **Instructions** sadrži sledeće metode:

* **static string getOpcode(char\* instruction, bool byte); -** na osnovu očitanog mnemonika i informacije da li je instrukcija jednobajtna ili dvobajtna vraća operacijski kod.
* **static bool is\_jump\_instruction(const char\* token); -** na osnovu očitanog mnemonika vraća indikator (logičkog tipa) da li instrukcija pripada instrukcijama skoka.

Klasa **Symbol\_table** implementiratabelu simbola u vidu ulančane liste čiji element predstavlja jedan red tabele i sadrži sledeće metode:

* **Element\* insert(string nm, string sec, bool def, int val, int vis, int t, Element\* n); -** ubacuje novi simbol u tabelu
* **int length(); -** vraća broj simbola u tabeli
* **Section\_info\* findSectionInfo(string name); -** na osnovu imena sekcije vraća informacije o sekciji
* **Element\* find\_by\_name(string name); -** dohvata red tabele na osnovu imena simbola/sekcije
* **Element\* find\_by\_ordinal(int ord); -** dohvata red tabele na osnovu rednog broja tog reda u tabeli
* **void write\_the\_table(); -** vršiispis tabele u tekstualni fajl.

Klasa **Relocation\_list** implementirarelokacionu listu za određenu sekciju. Svaka sekcija ima svoj objekat ove klase – listu zapisa o relokaciji i sadržaj. Metode ove klase su:

* **Element\* insert(const int& off, const string& t, const int& vr);** - ubacuje novi zapis u listu,
* **int length();** - vraća broj zapisa u listi,
* **void addByteToRLContent(char\* v); -** dodaje zadati bajt u sadržaj relokacione liste,
* **void addWordToRLContent(char\* v); -** dodaje zadatu reč u sadržaj relokacione liste,
* **void changeTheContent(char\* v, int LC, int size); -** menja sadržaj relokacione liste za zadati location counter, i u zavisnosti od argumenta size postavlja reč/bajt na vrednost zadatu drugim argumentom
* **void write\_the\_list(ofstream myfile); -** ispisuje listu zapisa o relokaciji i sadržaj u zadati fajl.

Klasa **One\_pass** implementirasam prolaz asemblera, kao i funkciju backpatch koja se izvršava nakon prolaza. Metode ove klase su:

* **void addSymbolToRL(char \*token, string type, int sz); -** dodaje simbol u relokacionu listu i odgovarajuće bajtove u sadržaj. Za nedefinisane simbole dodaje se „xx“ ili „xxxx“, što se kasnije razrešava backpatch metodom,
* **RLforSection\* getRL(string name); -** dohvata odgovarajuću relokacionu listu za zadato ime sekcije,
* **RLforSection\* addNewRL(string name); -** kreira novu relokacionu listu za zadato ime, i vraća pokazivać na nju,
* **int OperandSize(char\* token, bool dst, bool byte, bool jmps); -** na osnovu tokena koji predstavlja operand, informacije da li je operand destinacioni ili ne, širine instrukcije i informacije da li je instrukcija instrukcija skoka određuje tip adresiranja, dodaje odgovarajući sadržaj u odgovarajuću relokacionu listu i vraća broj bajtova koji je dodat u listu za sam operand, ne računajući prvi bajt koji predstavlja sam kod adresiranja
* **void pass(); -** ovaj metod je sam prolaz asebmblera, čita fajl red po red, razdvaja red na tokene i analizira tokene, proverava postojanje labele, direktive, instrucija, i poziva adekvatne metode koje obavljaju potreban posao, i prekida program u slučaju greške. Ako je neki simbol nedefinisan, pri svakom njegovom pojavljivanju pre nailaska na definiciju se u polje *flink,* koje je ulančana lista svih obraćanja tom simbolu pre definicije, dodaje novi element.
* **void process\_equ\_statement(string token, string sym\_name); -** procesira jedan izraz iz equ direktive, razdvaja ga na dva niza, niz operanada i niz operatora koji će kasnije poslužiti u izračunavanju izraza. U slučaju loše formulisanog izraza prijavljuje grešku i prekida program
* **void calculate\_equ\_values(); -** pokušava da razreši sve simbole definisane equ direktivom. Pošto su informacije o svakom izrazu sačuvane kao element ulančane liste, lista se prolazi najviše n puta (gde je n broj simbola definisanih putem equ) u pokušaju da se razreše svi simboli jer je pretpostavka da se u izrazu može naći i simbol koji je takođe definisan equ direktivom. U najgorem slučaju kada svaki izraz zavisi od nekog drugog simbola definisanog equ direktivom, treba nam više prolaza kroz listu (najviše n svakako) i zbog toga metoda i ima dve while petlje. Ako na kraju ostane nerazrešenih simbola javlja se greška.
* **void backpatch(); -** prolazi kroz listu simbola. Ako je neki simbol ostao nedefinisan prijavljuje grešku i prekida program. Ako neki simbol ima *flink* polje koje nije null, prolazi kroz celu listu *flink* i razrešava sva obraćanja unapred.
* **void writeRL(bool before); -** kreira izlazni fajl. Ukoliko je argument true ispiše sadržaj relokacionih lista za sve sekcije u fajl „before\_backpatch.txt“, a ako je argument false ispisuje sadržaj u fajl čiji je naziv zadat argumentom u komandnoj liniji. Deo sa before argumentom suštinski služi ako želimo da poredimo sadržaje lista pre backpatch metode sa finalnim sadržajem, da se ti sadržaji ne ispisuju u isti fajl.

U fajlu **prog.cpp** se nalazi main funkcija. Ona očitava argumente komandne linije, kreira objekat klase One\_pass za te argumente i pozove njenu metodu pass(), koja obavi prolaz kroz ulazni fajl. Zatim se za kreirani objekat poziva metod writeRL() sa argumentom true (ovo nije od suštinske važnosti jer ispisuje sadržaj relokacionih lista pre backpatchinga). Nakon toga se za isti objekat poziva metoda backpatch(), a potom ponovo writeRL() sa argumentom false. Ovaj poziv kreira zahtevani izlazni fajl.

Na kraju se poziva metoda koja ispisuje tabelu simbola u neki izlazni fajl. Ovaj ispis (koji i nije obavezan) mora se obavljati nakon poziva backpatch() jer se tokom backpatch menja tabela simbola budući da se razrešavaju oni koji su definisani equ direktivom, tako da je i besmisleno ispisivati tabelu pre toga jer je nepotpuna.

Sistem se prevodi komandom:

**g++ Convert.h Convert.cpp Instructions.h Instructions.cpp One\_pass.h One\_pass.cpp Relocation\_list.h Relocation\_list.cpp Symbol\_table.h Symbol\_table.cpp prog.cpp -std=c++11**

a pokreće komandom:

**./a.out ulaz1.s ulaz1.o**

Primer1:

.global a, c

.extern b, f

.text

jeq a

jeq e

jeq b

jeq d

jeq m

d: .word d

mov $b, %r0

mov %r0, c

mov $e, %r0

.data

.equ y, sara

m: .byte c

.skip 8

e: .word a

.word c

.equ z, y-8

.word bss

a: .word b

.bss

.equ x, a+7-c+0x76

.equ sara, x+2

c: .skip 8

.end

Tabela simbola:

#Tabela simbola

Redni broj | Ime | Sekcija | Vrednost | Velicina | Vidljivost | Tip

1 | UND | 1 | 0 | 0 | l | sekcija

2 | a | 10 | 15 | 0 | g | simbol

3 | c | 10 | 0 | 0 | g | simbol

4 | b | 1 | 0 | 0 | e | simbol

5 | f | 1 | 0 | 0 | e | simbol

6 | text | 6 | 0 | 37 | l | sekcija

7 | e | 10 | 9 | 0 | l | simbol

8 | d | 6 | 20 | 0 | l | simbol

9 | m | 10 | 0 | 0 | l | simbol

10 | data | 10 | 0 | 17 | l | sekcija

11 | y | 1 | 100 | 0 | l | equ

12 | z | 1 | 92 | 0 | l | equ

13 | bss | 10 | 0 | 8 | l | sekcija

14 | x | 1 | 98 | 0 | l | equ

15 | sara | 1 | 100 | 0 | l | equ

#end

Izlazni fajl:

#.ret.text

Offset|Tip|vrednost[text]:

6| R\_386\_16| 6

10| R\_386\_16| 1

14| R\_386\_16| 6

18| R\_386\_16| 6

20| R\_386\_16| 6

24| R\_386\_16| 1

34| R\_386\_16| 6

#.ret.data

Offset|Tip|vrednost[data]:

13| R\_386\_16| 10

15| R\_386\_16| 1

#.text

34 00 0F 00 34 00 09 00 34 00

00 00 34 00 14 00 34 00 00 00

14 00 64 00 00 00 20 64 20 80

00 00 64 00 09 00 20

#.data

00 00 00 00 00 00 00 00 00 0F

00 00 00 00 00 00 00

#.bss

00 00 00 00 00 00 00 00

#.end