REŠENJE

Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Programski Prevodioci 1 (IR4PP1)
Programski Prevodioci (RI4PP)
Nastavnik: doc. dr Dragan Bojić
Asistent: dipl. ing. Miloš Gligorić
Ispitni rok: Februar 2008.
Datum: 01.03.2008.

Napomene:

- *Ispit traje 3 sata. Prvih 30 minuta nije dozvoljeno napuštanje sale. Prvih sat i 30 minuta nije dozvoljeno iznositi ispitne zadatke. Poslednjih 10 minuta ispita nije dozvoljeno napuštanje sale.*

- *Na ispitnoj vežbanci obavezno naznačiti broj osvojenih poena na domaćem zadatku u sledećem formatu:*

  Rok u kome je odbranjen domaći _____/(npr: jun 2007)
  Ukupno poena _____/45

- *Na ispitnoj vežbanci potrebno je naznačiti pitanja i zadatke koji su radjeni (zaokruživanjem odgovarajćeg broja).*

- *Nepoštovanje nekog od pravila povlači negativne poene.*

Pitanja

1. [8] Objasniti podelu aktivnosti kod dvoprolaznih prevodilaca. Nabrojati prednosti i mane dvoprolaznih prevodilaca u odnosu na jednoprolazne.

2. [8] Definicija pravilne LR(1) konfiguracije. U kom odnosu sa LALR(1) parserom je pomenuta konfiguracija?

Zadaci

3. [11]

a. [2] Da li su dati automati ekvivalentni. Prikazati detaljan postupak.

	0	1	2	
Ax	Ax	Ax	Bx	0
Bx	Bx	Ax	Bx	1

	0	1	2	
Az	Cz	Az	Bz	0
Bz	Bz	Az	Az	1
Cz	Cz	Az	Bz	0

b. [5] Konvertovati regularni izraz: (b|ac)*jx* u nedeterministički automat (stanja automata obeležavati sa Sb gde b uzima vrednosti od 0). Koristiti Tompsonov algoritam.

c. [4] Nedeterministički automat dobijen pod b prebaciti u deterministički.

4. [4] Projektovati potisni automat koji prepoznaje sledeće skupove sekvenci: { 0^m 1^n} n > m >= 0. Objasniti.

5. [10]

a. [3] Dodeliti atribute sledećoj gramatici tako da neterminalni simbol <num> ima sintetizovan atribut jednak vrednosti oktalnog broja koji ovaj simbol generiše (iz oktalnog u decimalni sistem). Pretpostaviti da ulazni simbol OCT ima svoj atribut koji je jednak vrednosti odgovarajuće cifre (0, 1, 2, 3, 4, 5, 6, 7).

<num> → <num> OCT

<num> → OCT

b. [5] Implementirati TD parser na bazi rekurzivnog spusta za sledeću AT gramatiku:

1. $\langle integer \rangle_{value} \rightarrow DIGIT_{val}$ $\langle more\_digits \rangle_{v,\ b}$

value ← val * 10^b + v

2. $\langle more\_digits \rangle_{v,\ b} \rightarrow DIGIT_{val}$ $\langle more\_digits \rangle_{v1,\ b1}$

v ← val * 10^{b1} + v1 b ← b1 + 1

3. $\langle more\_digits \rangle_{v,\ b} \rightarrow \varepsilon$

v ← 0 b ← 0

c. [2] U prethodno napisan kod dodati oporavak od greške za neterminal <more_digits>. Oporavak treba da obuhvati ispis poruke korisniku i nastavak parsiranja. Nastavak parsiranja podrazumeva ponovno uparivanje neterminala <more_digits>.

6. [3] Za dati troadresni kod napisati implementaciju putem indirektnih trojki.

```
t1 := *p
t2 := i – 1;
b[t2] := t1;
t3 := j + 1;
j := t3;
t4 := a[j]
*p := t4;
```

7. [11] Dat je sledeći deo programa na jeziku sličnom Pascal-u:

```
procedure main;
        var ma, mb: integer;
        procedure procA(p1: integer, p2: integer);
              var a: integer;
              procedure procB;
              begin
                    procA(63, 23);
              end;
              procedure procC;
              begin
                    procB;
              end;
        begin
              procC;
        end;

begin
      mb := 4;
      procA(ma, 13);
end.
```

a. [2] Koliko je naredbi potrebno za pristup nelokalnoj promenljivoj putem displeja. Dati primer.

b. [5] Ako je statičko okruženje za nelokalne promenljive realizovano preko displeja, nacrtati izgled steka nakon formiranja aktivacionog zapisa za prvi poziv procA(63, 23). Dispelji se nalaze unutar aktivacionog zapisa. Za proceduru main kreira se aktivacioni zapis. Procedura main je na leksičkom nivou 1.

c. [4] Napisati 80x86 asemblerski potprogram ekvivalentan datom programu za slučaj b) ako se koriste naredbe enter i leave.

8. [5] Za dati Mikro Java bytecode treba napisati Mikro Java kod (.mj). Imena klasa trebaju da počinju slovom c, imena konstanti nizom karaktera con, imena globalnih promenljivih slovom g, imena lokalnih promenljivih i formalnih parametara slovom l, imana polja

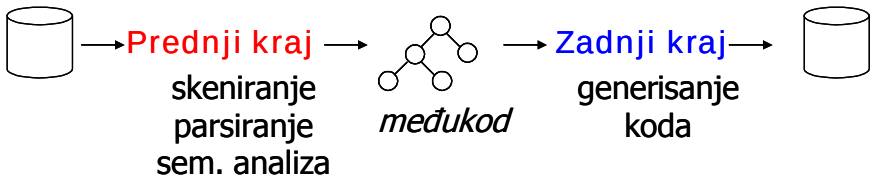
slovom f, imena metoda slovom m. Smatrati da se u programu koriste samo složeni tipovi i tip int. Za sve promenljive za koje nije poznat tip usvojiti int.

```
0. enter 2, 3
1. getstatic 1
2. const 53
3. add
4. load 0
5. add
6. putstatic 1
7. load 1
8. exit
9. return
10.enter 0,3
11.new 8
12.putstatic 2
13.getstatic 2
14.getfield 0
15.getstatic 2
16.getfield 1
17.add
18.store 2
19.exit
20.return
```

load b	... → ..., val
putstatic s	..., val → ...
getstatic s	... → ..., val
putfield s	..., adr, val → ...
const w	... → ..., val
add	..., val1, val2 → ..., val1+val2
new s	... → ..., adr
aload	..., adr, index → ..., val

Rešenja

1. [P1, slajd 33]



Prednosti

- Bolja prenosivost
- Moguće kombinovati različite prednje krajeve (zavisne od izvornog jezika) i zadnje (zavisne od ciljne mašinske arhitekture) čime se štedi na poslu izrade kompajlera za M jezika na N mašina.
- Optimizacije se lakše vrše na međukodu nego na izvornom kodu

Mane

- Sporije prevođenje
- Traži više memorije

2. [P6, strana 12]. LR(1) konfiguraciju <X> → α•β, t nazivamo pravilnom LR(1) konfiguracijom ako i samo ako je:

    1. t ∈ FOLLOW(<X>), ili
    2. reč o smeni 0 i t=ε.

U okviru stek simbola LALR(1) parsera mogu se pojaviti isključivo pravilne konfiguracije.

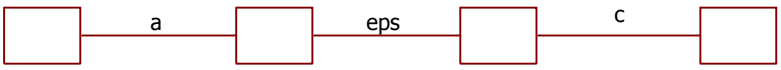
3.

a. Nisu

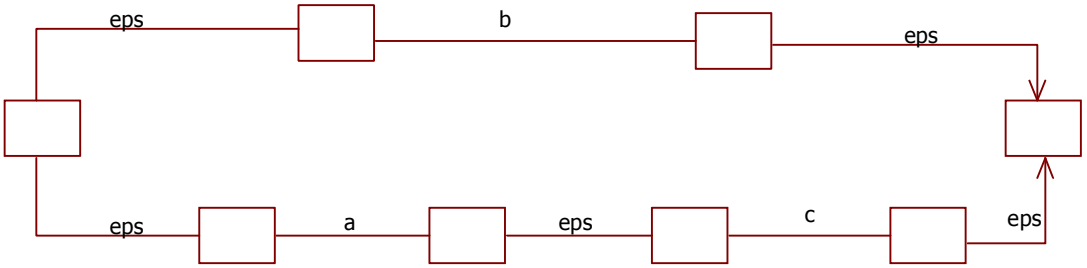
	0	1	2
Ax Az	Ax Cz	Ax Az	Bx Bz
Ax Cz	Ax Cz	Ax Az	Bx Bz
Bx Bz	Bx Bz	Ax Az	**Bx Az**

b.

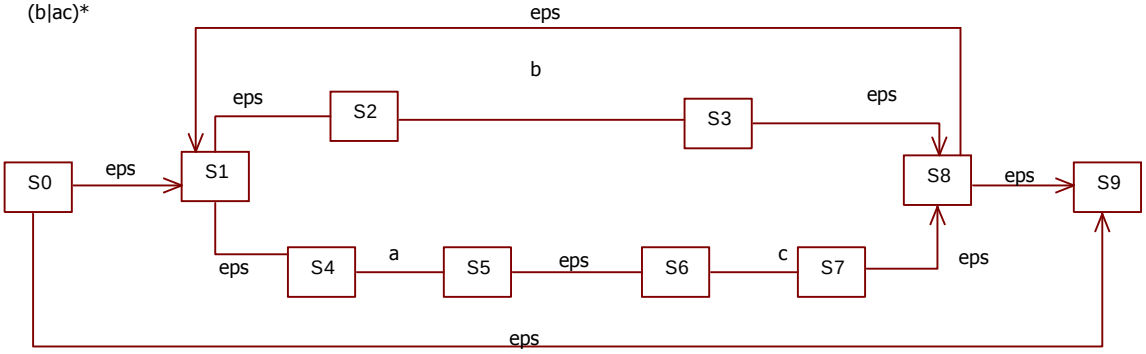
ac



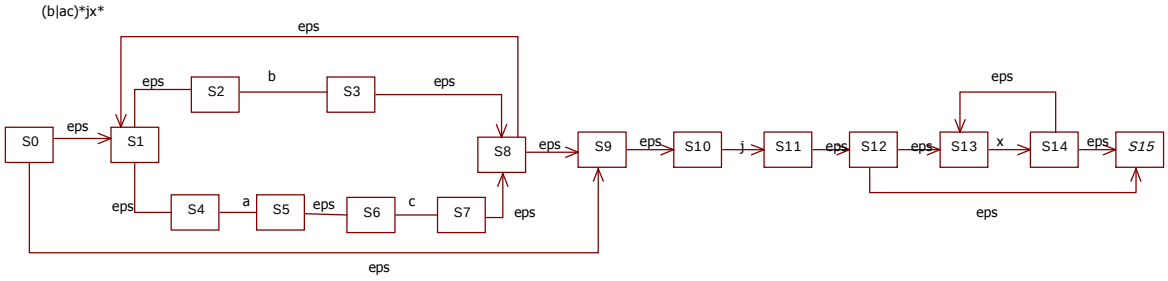
b|ac



(b|ac)*



(b|ac)*jx*



c.

	a	b	c	j	x	
S0 S1 S2 S4 S9 S10	S5 S6	S3 S8 S1 S2 S4 S9 S10		S11 S12 S13 S15		0
S5 S6			S7 S8 S1 S2 S4 S9 S10			0
S3 S8 S1 S2 S4 S9 S10	S5 S6	S3 S8 S1 S2 S4 S9 S10		S11 S12 S13 S15		0
S11 S12 S13 S15					S14 S13 S15	1

S7 S8 S1 S2 S4 S9 S10	S5 S6	S3 S8 S1 S2 S4 S9 S10		S11 S12 S13 S15		0
S14 S13 S15					S14 S13 S15	1

4.

STATE S1	0	1	-|
A	PUSH(A) ADVANCE STATE(S1)	ADVANCE STATE(S2)	REJECT
Δ	PUSH(A) ADVANCE STATE(S1)	ADVANCE STATE(S2)	REJECT

STATE S2	0	1	-|
A	REJECT	POP ADVANCE STATE(S2)	REJECT
Δ	REJECT	ADVANCE STATE(S2)	ACCEPT

5.

a.

$\langle num \rangle_{value} \rightarrow \langle num \rangle_{v}\ OCT_{c}$

value ← v * 8 + c

$\langle num \rangle_{value} \rightarrow OCT_{c}$

value ← c

b.

```
void PROC_integer (int &value)
{
int val = 0, v = 0; b = 0;
switch(current()){
 case DIGIT: val = current().IntValue();
             advance();
             PROC_digit_list(v, b);
             value = val * pow(10,b) + v;
             break;
 default:    error();
}
}

void PROC_digit_list (int &v, int &b)
{
int val = 0, v1 = 0, b1 = 0;
switch(current()){
```

```
 case DIGIT: val = current().IntValue();
             advance();
             PROC_digit_list(v1, b1);
             b = b1 + 1;
             v = val * pow(10, b1) + v1;
             break;
 case -|:    v = 0;
             b = 0;
             break;
default:     error();
}
}
```

c.

```
void PROC_digit_list (int &v, int &b)
{
int val = 0, v1 = 0, b1 = 0;
switch(current()){
 case DIGIT: val = current().IntValue();
             advance();
             PROC_digit_list(v1, b1);
             b = b1 + 1;
             v = val * pow(10, b1) + v1;
             break;
 case -|:    v = 0;
             b = 0;
             break;
default:     print("greska u ulazu");
             advance();
             PROC_digit_list(v, b);
             break;
}
}
```

6.

Adresa	Operacija	Operand1	Operand2
Mem1	assign*	Mem1	p
Mem2	-	i	1
Mem3	[]=	b	Mem2
Mem4	assign	Mem3	Mem1
Mem5	+	j	1
Mem6	assign	j	Mem5
Mem7	=[]	a	j
Mem8	assign	Mem8	Mem7
Mem9	*assign	p	Mem8

Indeks	Pokazivač
(0)	Mem1
(1)	Mem2
(2)	Mem3
(3)	Mem4
(4)	Mem5
(5)	Mem6
(6)	Mem7
(7)	Mem8
(8)	Mem9

7.

a. Za pristup nelokalnoj promenljivoj putem displeja uvek su dovoljne dve naredbe, bez obzira na kojem se leksičkom nivou nalazi nelokalna promenljiva: Pva naredba u registar dovodi sadržaj odgovarajueg pokazivača koji predstavlja adresu aktivacionog zapisa u kome se nalazi nelokalna promenljiva. Druga naredba vrši pristup koristeći bazno indeksiranje sa poznatim pomerajem.

b.

Povratna adresa main
Kontrolna veza
D[1] = main
ma
mb
ma
13
Povratna adresa procA(ma, 13)
Kontrolna veza
D[1] = main
D[2] = procA(ma, 13)
a
Povratna adresa procC
Kontrolna veza
D[1] = main
D[2] = procA(ma, 13)
D[3] = procC
Porvatna adresa procB
Kontrolna veza
D[1] = main
D[2] = procA(ma, 13)

D[3] = procB
63
23
Kontrolna veza
D[1] = main
D[2] = procA(62, 23)
a

c.

```
; kod za main

; postoje 2 lokalne promenljive, a leksicki nivo je jednak 1
ENTER 4, 1
MOV [BP-6], 4    ; mb = 4
PUSH [BP-4]
MOV AX, 13
PUSH AX
; poziv procedure procA
CALL procA
; sekvenca povratka
LEAVE
RET

; kod za proceduru procA

; procA ima jednu lokalnu promenljivu i nalazi se na leksickom
; nivou 2
ENTER 2, 2
; poziva se procC
CALL procC
; sekvenca povratka
LEAVE
; pri povratku se skidaju stvarni parametri sa steka
RET 0004

; kod za proceduru procC

; procC nema lokalne promenljive i nalazi se na leksickom
; nivou 3
ENTER 0, 3
; poziva se procB
CALL procB
; povratak u pozivajucu proceduru
LEAVE
RET

; kod za proceduru procB

; procB nema lokalne promenljive i nalazi se na leksickom
; nivou 3
ENTER 0, 3
; prvi parametar za poziv procA se prvo stavi u AX
MOV AX, 63
; stavljanje prvog parametra za poziv procA na stek
```

```
PUSH AX
; drugi parametar za poziv procA se prvo stavi u AX
MOV AX, 23
; stavljanje drugog parametra za poziv procA na stek
PUSH AX
CALL procA
LEAVE
RET
```

8.

```
class c0
  final int con0 = 2;
  final int con1 = 53;
  int g0, g1;

  class c1 {
    int f0, f1;
  }

  c1 g2;
{
  int m0(int l0, int l1)
    int l2;
  {
    g1 = g1 + con1 + l0;
    return l1;
  }

  void m1()
    int l0, l1, l2;
  {
    g2 = new c1;
    l2 = g2.f0 + g2.f1;
  }
}
```