Fakultet elektrotehnike i računarstva
Sveučilišta u Zagrebu

Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave

Daniel Skrobo
Ivan Žužak
Miroslav Popović

Prevođenje programskih jezika

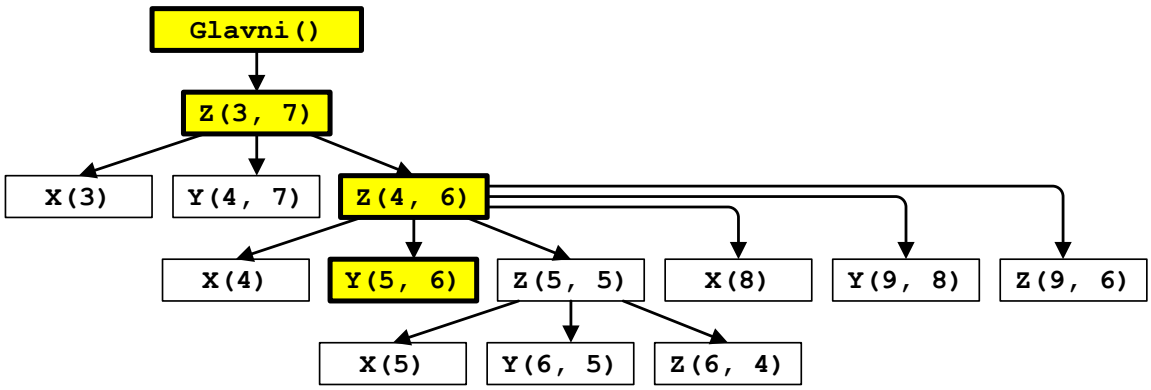
Auditorne vježbe

Zagreb, rujan 2007.

15. Za zadani program izgradite stablo aktiviranja procedura.

```
01 Glavni()
02    X(a)
03    {
04       vrati a + 1;
05    }
06    Y(b, c)
07    {
08       vrati c - b/4;
09    }
10
11    Z(d, e)
12    {
13       dok (d <= e)
14       {
15          d = X(d);
16          e = Y(d, e);
17          Z(d, e);
18          if (d == 5)
19          {
20             d = 8;
21             e = 8;
22             dalje;
23          }
24       }
25    }
26 {
27    Z(3, 7)
28 }
```

```
Zap Glavni()
   Zap Z(3, 7)
      Zap X(3)
      Zav X(3)
      Zap Y(4, 7)
      Zav Y(4, 7)
      Zap Z(4, 6)
         Zap X(4)
         Zav X(4)
   ⟶    Zap Y(5, 6)
         Zav Y(5, 6)
         Zap Z(5, 5)
            Zap X(5)
            Zav X(5)
            Zap Y(6, 5)
            Zav Y(6, 5)
            Zap Z(6, 4)
            Zav Z(6, 4)
         Zav Z(5, 5)
         Zap X(8)
         Zav X(8)
         Zap Y(9, 8)
         Zav Y(9, 8)
         Zap Z(9, 6)
         Zav Z(9, 6)
      Zav Z(4, 6)
   Zav Z(3, 7)
Zav Glavni()
```

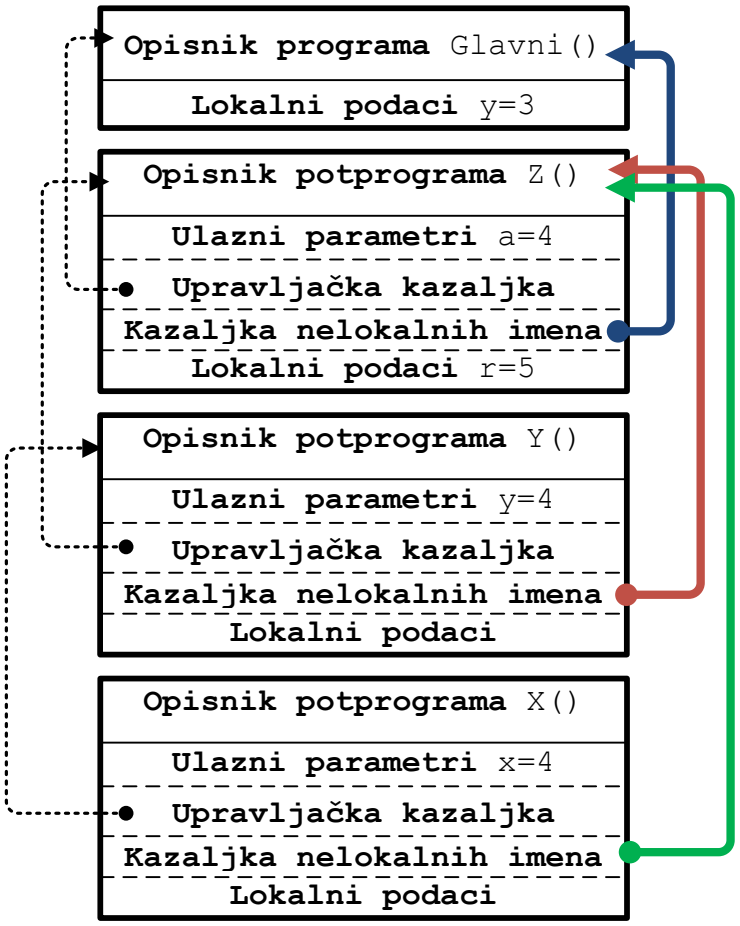


Stablo aktiviranja procedura zadanog programa

16. Za zadani program prikažite sadržaj opisinka procedura u trenutku prije izvođenja naredbe 07 ako se koristi: (*a*) *statičko pravilo djelokruga*, (*b*) *dinamičko pravilo djelokruga*.

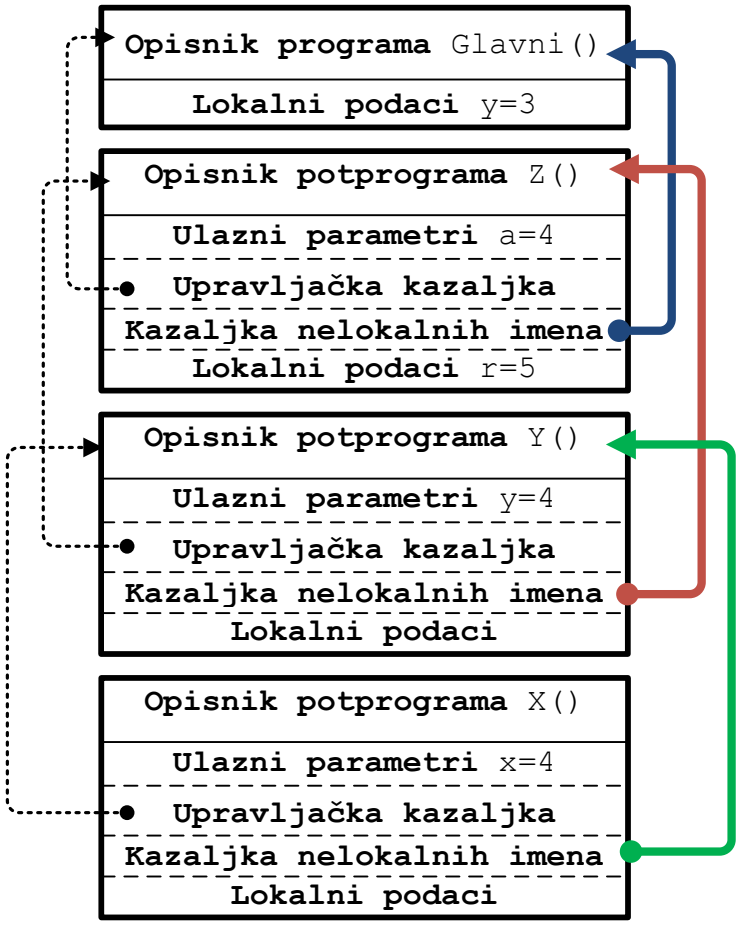
a) Statičko pravilo djelokruga (ugnijžđenih procedura)

```
01 Glavni()
02    int y = 3;
03    def Z(a)
04       int  r = 5
05       def X(x)
06       {
07          vrati x+y+1;
08       }
09       def Y(y)
10       {
11          vrati X(y)+1;
12       }
13    {
14       Y(a);
15    }
16 {
17    ispiši Z(y+1)
18 }
```



b) Dinamičko pravilo djelokruga

```
01 Glavni()
02    int y = 3;
03    def Z(a)
04       int  r = 5
05       def X(x)
06       {
07          vrati x+y+1;
08       }
09       def Y(y)
10       {
11          vrati X(y)+1;
12       }
13    {
14       Y(a);
15    }
16 {
17    ispiši Z(y+1)
18 }
```



17. Prikazati razliku između razmjene parametara primjenom *mehanizma razmjene adresa* i *mehanizma razmjene imena* na sljedećem programu:

```
01 var x = 0
02 polje A = {10, 20} // A[0]=10, A[1]=20
03 P(a) {
04   x = 1
05   a = 100
06   Ispisi(A[0], A[1])
07 }
08 {
09   P(A[x])
10 }
```

a) Razmjena adresa

- poziv procedure *P* u retku 09 pridružuje parametru ***a*** adresu od ***A[x] = A[0]*** (vrijednost varijable ***x*** je 0 u trenutku poziva procedure)
- naredba 04 mijenja vrijednost varijable ***x*** u 1 (ne utječe na ***a***)
- naredba 05 mijenja ***A[0]*** u 100
- naredba 06 ispisuje 100 20

b) Razmjena imena

- poziv procedure *P* u retku 09 pridružuje parametru ***a*** ime ***"A[x]"***
- naredba 04 mijenja vrijednost varijable ***x*** u 1
- naredba 05 ekvivalentna je naredbi ***A[x] = 100***
    - s obzirom da je vrijednost varijable ***x*** 1, naredba 05 pridružuje vrijednost 100 lokaciji ***A[1]***
- naredba 06 ispisuje 10 100

18. Za zadani program prikažite vrijednosti globalnih i lokalnih varijabli tijekom izvođenja programa. Razmjena parametara procedura ostvaruje se primjenom *mehanizma razmjene imena*.

```
01  varijabla x=0, y=3, z=-1;
02  polje o = {0, 0, 0, 10, 20};
03  Racunaj(p, q, r) {
04     z = p + x;
05     z = (q + 1) % 2 + 3;
06     Ispisi(p, x, r);
07     r = z + q;
08  }
09  {
10     za x = 3 do 4 {
11        Racunaj(o[x], o[3+x%2], z);
12        Ispisi(x, y, z, o[3], o[4]);
13     }
14  }
```

```
01  varijabla x=0, y=3, z=-1;
02  polje o = {0, 0, 0, 10, 20};
03  Racunaj(p, q, r) {                          II
04     z = p + x;
05     z = (q + 1) % 2 + 3;
06     Ispisi(p, x, r);                         III
07     r = z + q;                               IV
08  }
09  {                                           I
10     za x = 3 do 4 {
11        Racunaj(o[x], o[3+x%2], z);
12        Ispisi(x, y, z, o[3], o[4]);          V
13     }
14  }
```

Ugradnja kontrolnih točaka na kojima promatramo stanje programa u izvođenju

Stanje na početku izvođenja programa:

	x	y	z	o[3]	o[4]	p	q	r
I	0	3	-1	10	20			

Započinje se s izvođenjem prve iteracije petlje i poziva se potprogram *Racunaj*. Stanje na početku izvođenja potprogama:

	x	y	z	o[3]	o[4]	p	q	r
II	3	3	-1	10	20	10	20	-1

Izvode se naredbe:

```
04  z = p + x;
05  z = (q + 1) % 2 + 3;
06  Ispisi(p, x, r);
```

Na sljedeći način:

```
04  z = o[x] + x;
          → z = 10 + 3 → z = 13 (r = 13)
05  z = (o[3+x%2] + 1) % 2 + 3;
          → z = (20 + 1) % 2 + 3 → z = 4 (r = 4)
06  Ispisi(p, x, r);
```

Stanje nakon izvođenja navedenih naredbi:

	x	y	z	o[3]	o[4]	p	q	r
III	3	3	**4**	10	20	10	20	**4**
Ispis	**10, 3, 4**							

Izvodi se naredba:

```
07  r = z + q;
```

Na sljedeći način:

```
07  r = z + o[3+x%2];
```

→ *r = 4 + 20* → ***r = 24 (z = 24)***

Stanje nakon izvođenja navedene naredbe:

	x	y	z	o[3]	o[4]	p	q	r
IV	3	3	**24**	10	20	10	20	**24**

Izlazi se iz potprograma i izvodi naredba:

```
07  Ispisi(x, y, z, o[3], o[4]);
```

Ispis nakon izvođenja navedene naredbe:

Ispis	**3, 3, 24, 10, 20**

Započinje se s izvođenjem <u>druge</u> iteracije petlje i poziva se potprogram *Racunaj*. Stanje na početku izvođenja progama:

	x	y	z	o[3]	o[4]	p	q	r
II	**4**	3	24	10	20			

Izvode se naredbe:

```
04  z = p + x;
05  z = (q + 1) % 2 + 3;
06  Ispisi(p, x, r);
```

Na sjedeći način:

```
04  z = o[x] + x;
```

→ z = 20 + 4 → **z = 24**

```
05  z = (o[3+x%2] + 1) % 2 + 3;
```

→ z = (10 + 1) % 2 + 3 → **z = 4**

```
06  Ispisi(p, x, r);
```

Stanje nakon izvođenja navedenih naredbi:

	x	y	z	o[3]	o[4]	p	q	r
III	4	3	**4**	10	20	20	10	**4**
Ispis	**20, 4, 4**							

Izvodi se naredba:

```
07  r = z + q;
```

Na sljedeći način:

```
07  r = z + o[3+x%2];
          → r = 4 + 10 → r = 14
```

Stanje nakon izvođenja navedene naredbe:

	x	y	z	o[3]	o[4]	p	q	r
IV	4	3	**14**	10	20	20	10	**14**

Izlazi se iz potprograma i izvodi naredba:

```
07  Ispisi(x, y, z, o[3], o[4]);
```

Ispis nakon izvođenja navedene naredbe:

Ispis	**4, 3, 14, 10, 20**

Svi ispisi:

Prvi ispis: **10, 3, 4**
Drugi ispis: **3, 3, 24, 10, 20**
Treći ispis: **20, 4, 4**
Četvrti ispis: **4, 3, 14, 10, 20**

19.Izgradite atributnu prijevodnu gramatiku koja generira troadresne naredbe za računanje logičkih izraza koji sadrže operator $\wedge$, $\vee$ i $\neg$.

a) Izgradnja gramatike za parsiranje logičkih izraza

(1) <S> → <E>

(2) <E> → ¬ (<E>)

(3) <E> → <E> ∧ <E>

(4) <E> → <E> ∨ <E>

(5) <E> → *laž*

(6) <E> → *istina*

b) Proširivanje gramatike atributima

(1) $\langle S \rangle_{kod} \rightarrow \langle E \rangle_{ime1,\ kod1}$

(2) $\langle E \rangle_{ime1,\ kod1} \rightarrow \neg\ (\ \langle E \rangle_{ime2,\ kod2}\ )$

(3) $\langle E \rangle_{ime1,\ kod1} \rightarrow \langle E \rangle_{ime2,\ kod2} \wedge \langle E \rangle_{ime3,\ kod3}$

(4) $\langle E \rangle_{ime1,\ kod1} \rightarrow \langle E \rangle_{ime2,\ kod2} \vee \langle E \rangle_{ime3,\ kod3}$

(5) $\langle E \rangle_{ime1,\ kod} \rightarrow laž_{\ ime2}$

(6) $\langle E \rangle_{ime1,\ kod} \rightarrow istina_{\ ime2}$

c) Proširivanje gramatike akcijskim znakovima

(1) <S>$_{kod}$ → <E> $_{ime1,\ kod1}$
{ Kod = Kod1; }

(2) <E> $_{ime1,\ kod1}$ → ¬ (<E> $_{ime2,\ kod2}$)
{
 Ime1 = NovoIme();
 Kod1 = Generiraj(kod2 || Ime1 ":= not " Ime2);
}

(3) <E> $_{ime1,\ kod1}$ → <E> $_{ime2,\ kod2}$ ∧ <E> $_{ime3,\ kod3}$
{
 Ime1 = NovoIme();
 Kod1 = Generiraj(kod2 || kod3 || Ime1 ":=" Ime2 "and" Ime3);
}

(4) <E> $_{ime1,\ kod1}$ → <E> $_{ime2,\ kod2}$ ∨ <E> $_{ime3,\ kod3}$
{
 Ime1 = NovoIme();
 Kod1 = Generiraj(kod2 || kod3 || Ime1 ":=" Ime2 "or" Ime3);
}

(5) <E> $_{ime1,\ kod1}$ → *laž*
{ Ime1 = 0; Kod1 = Generiraj(""); }

(6) <E> $_{ime1,\ kod1}$ → *istina*
{ Ime1 = 1; Kod1 = Generiraj(""); }

20. Za zadani program izgradite graf tijeka izvođenja.

```
01        Input(n)
02        Input(p)
03        a0 := 2
04        if n <= 5 goto L1
05        if p > 5 goto L2
06  L1:   a1 := a0 + 3
07        a2 := a1 + p
08        a3 := a1 * n
09        goto Z
10  L2:   a2 := 3 * 3
11        Output (a3)
12        p := p + 1
13        goto L1
14  Z:    nop
```