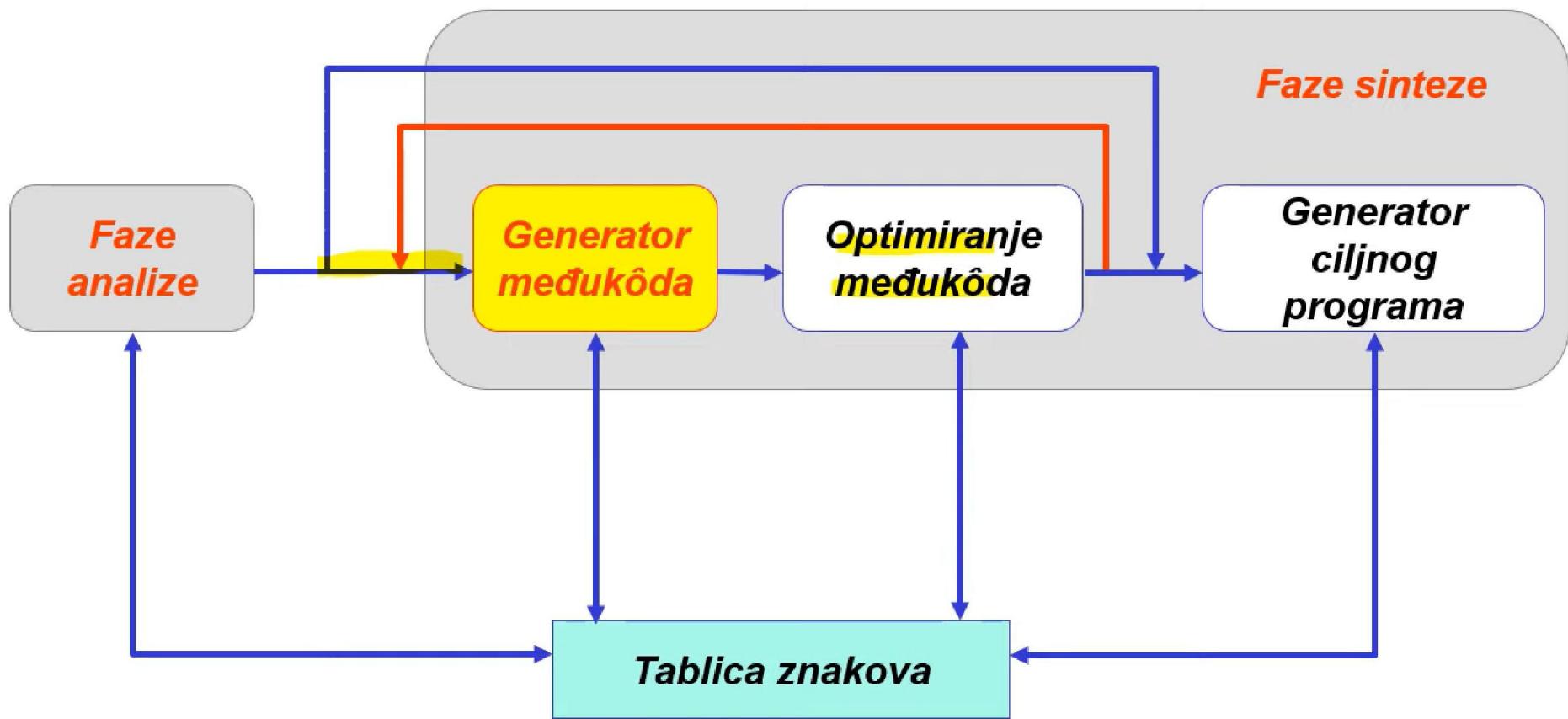


Generiranje međukôda



Završava analiza izvornog i započinje sinteza ciljnog programa

Međukôd je prijelazni oblik izvornog programa koji se koristi tijekom sinteze ciljnog programa

Generiranje međukôda

- **Generiranje međukôda omogućuje:**
 - odvojeno i nezavisno programsko ostvarenje faza analize i sinteze
 - učinkovito optimiranje

Generiranje međukôda

- Odvojeno i nezavisno programsko ostvarenje faza analize i sinteze

- **Analizator izvornog jezika**

- moguće je iskoristiti u jezičnim procesorima koji generiraju strojne jezike računala različitih arhitektura

- **Program sinteze ciljnog programa**

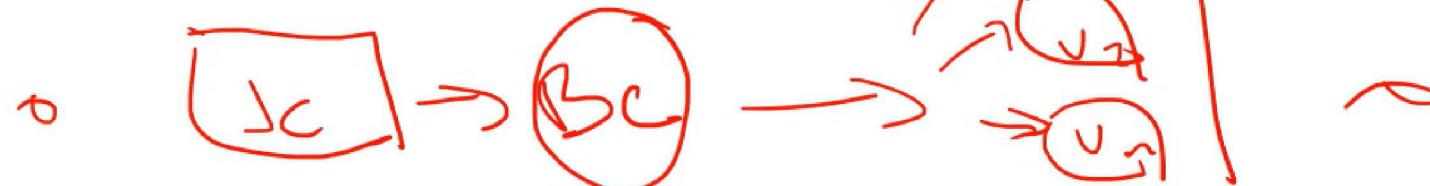
- moguće je iskoristiti u jezičnim procesorima različitih izvornih jezika

- **P-kôd**

- jezik virtualnog stogovnog stroja
 - međukôd jezičnog procesora programskog jezika Pascal ←
 - međukôd jezičnog procesora programskog jezika Modula-2 ←

- **JAVA ByteCode**

- jezik ~~Java~~ virtualnog stroja



Generiranje međukôda

• Optimiranje

- **Razvijeno je mnoštvo postupaka optimiranja**
 - struktura izvornog programa
 - programske petlje, aritmetički izrazi, procedure
 - struktura strojnog programa
 - izbor strojne naredbe, redoslijed izvođenja strojnih naredbi, izbor mesta dohvata vrijednosti operanda
 - uporaba sredstava računala
 - izbor registara, izbor aritmetičko-logičke jedinice, upravljanje memorijskom hijerarhijom
- **Izbor strukture međukôda ovisi o:**
 - izvornom jeziku
 - ciljnom jeziku
 - arhitekturi računala
 - vrsti postupka optimiranja

Generiranje međukôda

- **Struktura međukôda**

- **Razina**

- omjer zastupljenosti struktura izvornog i ciljnog jezika

- **Oblik**

- linearni

- postfiksni

- grafički

Razine međukôda

- **Osnovne razine međukôda**
 - **Međukôd više razine**
 - **Međukôd srednje razine**
 - **Međukôd niže razine**

Međukôd više razine

- Započinje sinteza ciljnog programa
- Ostaju sačuvane
 - strukture petlji i indeksa polja izvornog programa
- Međukôd čine
 - naredbe pridruživanja, bezuvjetnog grananja, uvjetnog grananja, naredbe petlji i upravljanja procedurama
 - naredbe upravljanja tijekom izvođenja programa koriste simboličke programske oznake
 - koriste se naredbe slične strojnim naredbama
 - na primjer naredba poziva programske stupice
- Pogodan je za različite vrste analiza
 - analiza tijeka izvođenja programa, analiza toka podataka, analiza zavisnosti podataka i analiza pseudonima
- Grafički oblici
 - sažeto sintaksno stablo, usmjereni graf bez petlji i graf zavisnosti programa

Međukôd više razine

Ime := *Popis*[*i*, *j*+7]

Međukôd srednje razine

- **Međukôd čine**
 - pojednostavljene naredbe izvornog jezika koje sliče strojnim naredbama
 - jednostavne naredbe uvjetnog i bezuvjetnog grananja
 - naredbe poziva i povratka iz potprograma
 - simbolička imena identifikatora izvornog programa
 - simbolička imena privremenih varijabli
 - optimiranje
 - izvođenje procedura, tijek izvođenja programa, redoslijed izvođenja naredbi i računanje izraza

Međukôd srednje razine

■ ***Ime := Popis[i, j+7]***

- 1) $p1 := j + 7$
- 2) $p2 := i * 31$
- 3) $p3 := p1 + p2$
- 4) $p4 := 2 * p3$
- 5) $p5 := \text{Adresa}(Popis[])$
- 6) $p6 := p5 + p4$
- 7) $\text{Ime} := *p6$

Međukôd niže razine

- **Naredbe međukôda niže razine**
 - **slične naredbama strojnog jezika računala**
 - **prevodenje u strojne naredbe**
 - jedan u jedan
 - **izbor:**
 - najpovoljnije strojne naredbe
 - najpovoljniji način adresiranja
 - **ne koriste se simbolička imena identifikatora**
 - vrijednosti se dohvaćaju primjenom registara ili memorijskih adresa
 - simbolička imena registara
 - neograničen broj simboličkih registara
- **postupci strojno zavisnog optimiranja**
 - izbor strojnih naredbi, dodjela registara, izbor redoslijeda izvođenja strojnih naredbi

Međukôd niže razine

→ $Ime := Popis[i, j+7]$

- 1) → $p1 := j + 7$
- 2) $p2 := i * 31$
- 3) $p3 := p1 + p2$
- 4) $p4 := 2 * p3$
- 5) $p5 := \text{Adresa}(Popis[])$
- 6) $p6 := p5 + p4$
- 7) $Ime := *p6$

- 1) $r1 \leftarrow (\text{PočetakStatičkeMemorije})$
- 2) $r2 \leftarrow r1 + 7$
- 3) $r3 \leftarrow (\text{PočetakStatičkeMemorije} + 2)$
- 4) $r4 \leftarrow r3 * 31$
- 5) $r5 \leftarrow r4 + r2$
- 6) $r6 \leftarrow 2 * r5$
- 7) $r7 \leftarrow \text{PočetakStatičkeMemorije} + 6$
- 8) $r8 \leftarrow r7 + r6$
- 9) $r9 \leftarrow \text{PočetakStatičkeMemorije} + 4$
- 10) $(r9) \leftarrow (r8)$

→ $\text{PočetakStatičkeMemorije}$
 $\text{PočetakStatičkeMemorije} + 2$
 $\text{PočetakStatičkeMemorije} + 4$
 $\text{PočetakStatičkeMemorije} + 6$
 $\text{PočetakStatičkeMemorije} + 8$

j	←
i	←
Ime	←
Prvi element polja $Popis[]$	
Drugi element polja $Popis[]$	

Oblici međukôda

- **Oblici međukôda**

- **Grafički**

- sažeto sintaksno stablo
 - usmjereni graf bez petlji

- **Postfiksni**

- **P-kôd**
 - **JAVA ByteCode**

- **Linearni**

- troadresne naredbe

- **Posebni oblici međukôda**

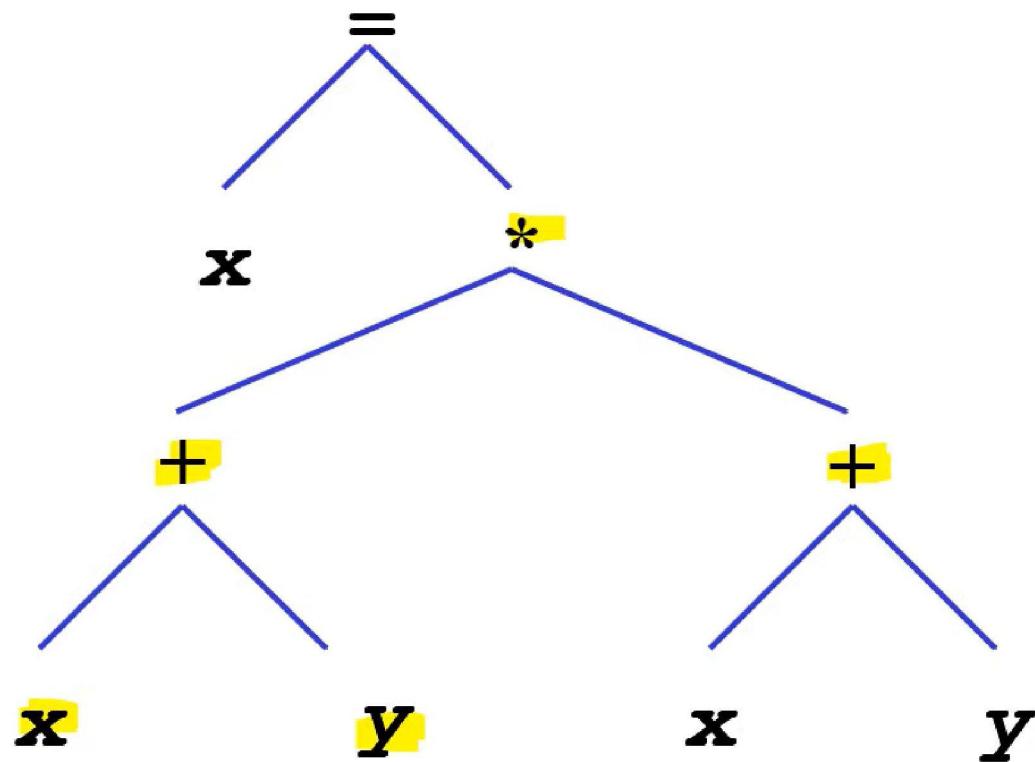
- statičko jednostruko pridruživanje
 - graf zavisnosti

Grafički oblici međukôda

- **Sažeto sintaksno stablo**
 - **Operandi**
 - listovi sažetog sintaksnog stabla
 - **Operatori i ključne riječi**
 - unutrašnji čvorovi

Sažeto sintaksno stablo

$$x = (x+y) * (x+y)$$

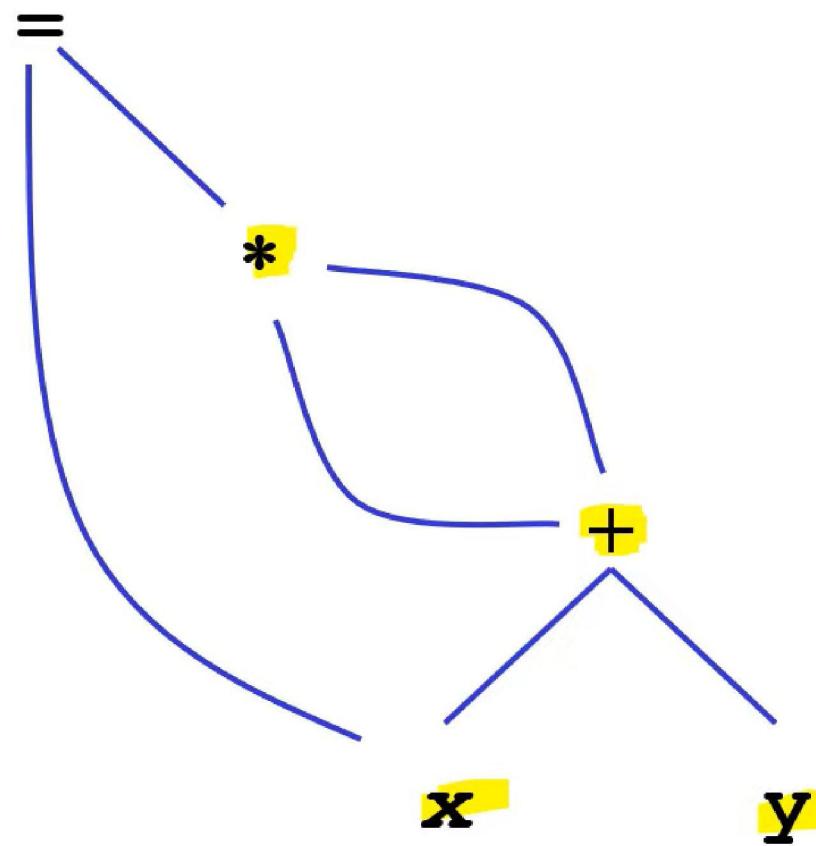


Grafički oblici međukôda

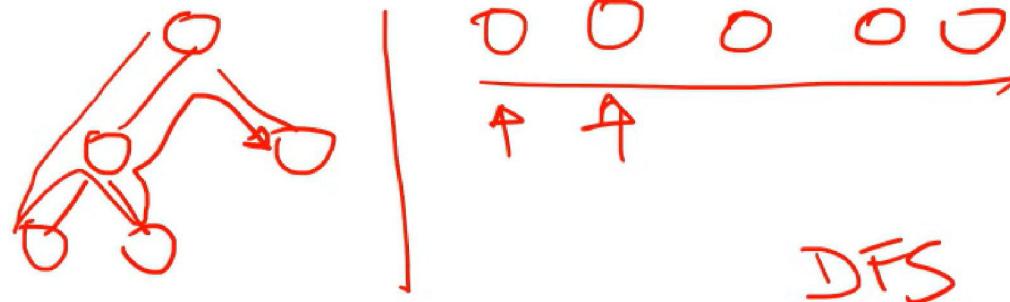
- Usmjereni graf bez petlji
 - Istovjetni izrazi prikazuju se jednim podstablom

Izravni graf bez petlji

$$\textcolor{yellow}{x} = (\textcolor{yellow}{x+y}) * (\textcolor{yellow}{x+y})$$

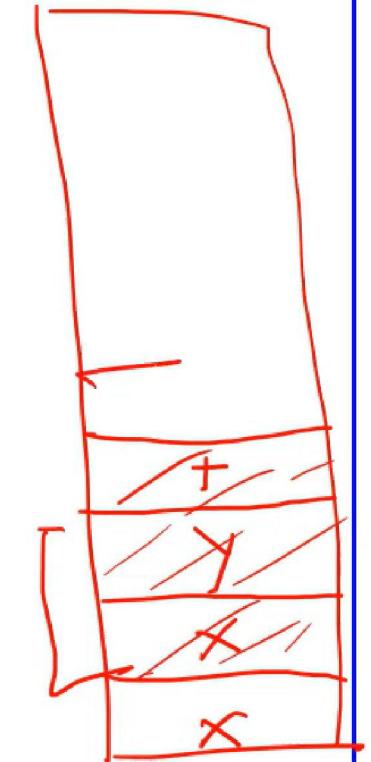
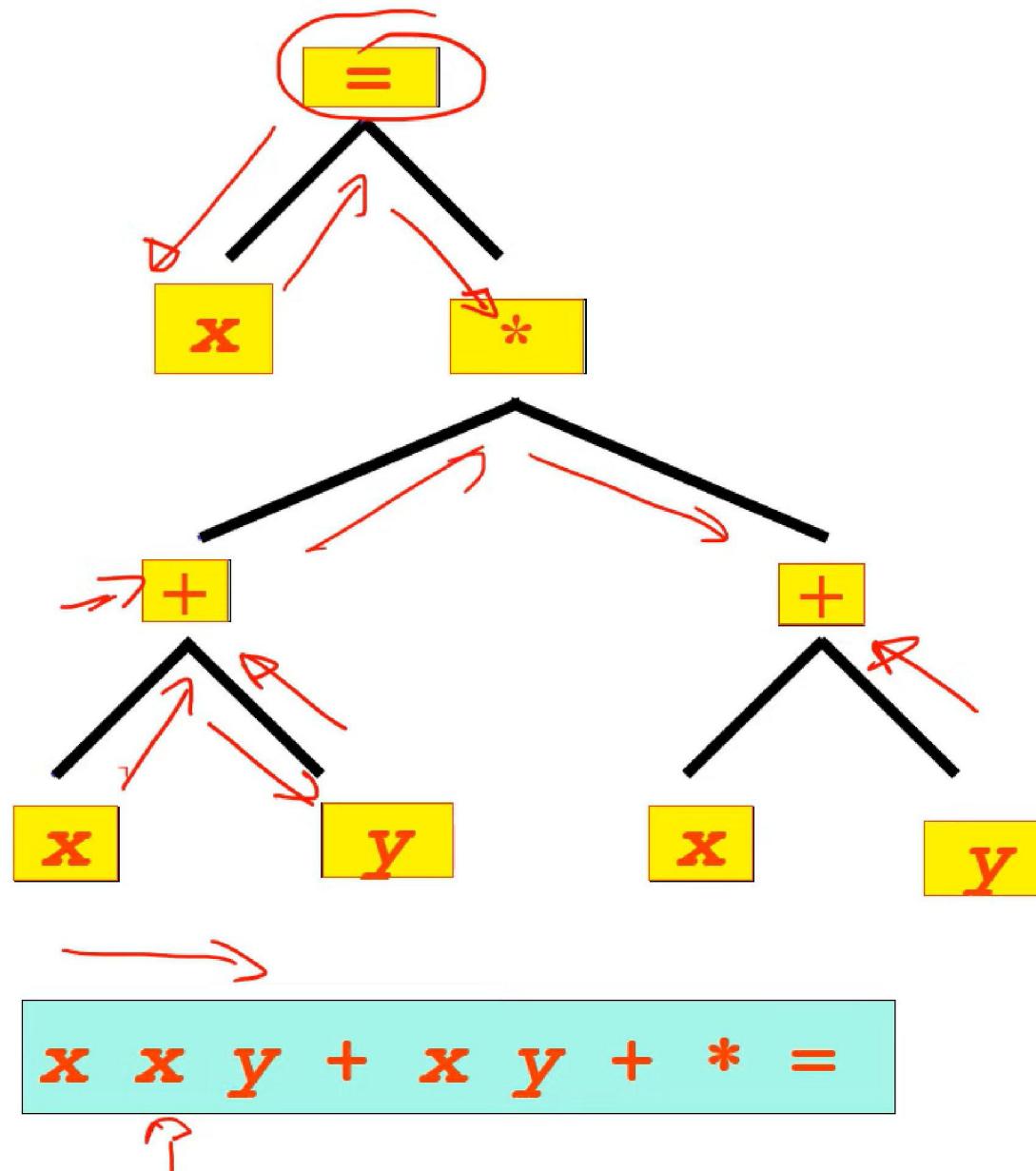


Postfiksni oblik međukôda



- **Nastaje**
 - obilaskom čvorova sažetog sintaksnog stabla po dubini
- **U usporedbi s infiksnim sustavom oznaka**
 - koristi manji memorijski prostor
- **Potpuno izravnavanje sintaksnog stabla**
 - postiže se primjenom potisnog stoga
- **Tijekom postupka izravnavanja**
 - na potisni stog spremaju se operandi i operacije

Postfiksni oblik međukôda



P-kôd

- Izvođenje
 - virtualni stogovni stroj
- Međukôd jezičnih procesora programskih jezika
 - Pascal i Modula-2
 - prvotne inačice jezičnog procesora programskog jezika C
 - prva osobna računala imala su male radne memorije
 - strojne naredbe Intelovih procesora zahtijevale su daleko veći memorijski prostor od naredbi P-kôda

• Naredbe P-kôda

- veličina
 - četiri okteta
- prvi oktet
 - operacijski kôd
 - devet osnovnih operacijskih kôdova
- preostala tri okteta
 - različito značenje ovisno o operacijskom kôdu

P-kôd

LIT 0,NN

Na vrh stoga stavi se konstanta NN

LOD 255,0

S vrha stoga uzme se adresa. Primjenom pročitane adrese dohvati se vrijednost koja se zapiše na vrh stoga

STO 255,0

S vrha stoga uzme se vrijednost i adresa. Pročitana vrijednost zapiše se primjenom pročitane adrese

OPR 0,2

S vrha stoga uzmu se dvije vrijednosti. Na vrh stoga stavi se njihov zbroj

OPR 0,5

S vrha stoga uzmu se dvije vrijednosti. Na vrh stoga stavi se njihov količnik

OPR 0,15

S vrha stoga uzmu se dvije vrijednosti. Na vrh stoga stavi se rezultat njihove logičke i-operacije

OPR 0,17

S vrha stoga uzmu se dvije vrijednosti. Na vrh stoga stavi se druga pročitana vrijednost s pomaknutim bitovima. Veličina pomaka zadaje se prvom pročitanom vrijednošću.

P-kôd

OPR 0,19

S vrha stoga uzme se vrijednost. Na vrh stoga stavi se pročitana vrijednost uvećana za jedan.

OPR 0,21

S vrha stoga uzme se vrijednost. Na vrh stoga stavi se pročitana vrijednost dva puta.

CSP 0,0

Pročitana vrijednost ulaza stavi se na vrh stoga.

CSP 0,1

S vrha stoga uzme se vrijednost i ispiše se na izlaz.

JMP 0,A

Bezuvjetni skok na naredbu adrese A.

CAL 255,0

S vrha stoga uzme se adresa, a na stog se stavi povratna adresa potprograma. Izvođenje programa nastavlja se adresom pročitanom s vrha stoga.

JAVA ByteCode

- Izvodi se primjenom JAVA virtualnog stroja
- JAVA virtualni stroj
 - Skup naredbi
 - Skup registara
 - Područje metode
 - Gomila
 - Stog

- **Naredbe Java virtualnog stroja**
 - mnemoničkog oblika
 - **operacijski kôd naredbe**
 - veličine jednog okteta
 - **podatkovni objekti**
 - oktet, kratka riječ, duga riječ, znakovni podatak, cijeli brojevi različite veličine i brojevi s posmačnim zarezom različite preciznosti
 - **operacije nad različitim podatkovnim objektima**
 - **iadd** zbraja cijele brojeve
 - **ladd** zbraja dva duga cijela broja
 - **fadd** zbraja dva broja s posmačnim zarezom
 - **dadd** zbraja dva broja s posmačnim zarezom dvostrukе preciznosti

- **Registri**

• Osprema se stanje Java virtualnog stroja

- registri su veličine **32 bita**

- **četiri registra**

- **pc** - programsko brojilo

- **optop** - kazaljka koja pokazuje na vrh stoga operanada

- **frame** - kazaljka koja pokazuje na okolinu metode
koja se u tom trenutku izvodi

- **vars** - kazaljka koja pokazuje na lokalne varijable metode

• vrijednosti se razmjenjuju primjenom stoga, a ne primjenom registara

JAVA ByteCode

- **Područje metode**
 - prostor u koji se sprema ciljni program metode, tablica znakova
- **Gomila**
 - koristi se za dinamičku dodjelu memorijskog prostora
 - objektima se pristupa primjenom kazaljki koje pokazuju na dodijeljeni memorijski prostor

JAVA ByteCode

- **Stog**
 - aktiviranim metodama dodjele se **okviri**
 - uloga okvira metode slična je ulozi opisnika procedure
 - **okvir čine tri dijela**
 - lokalne varijable**
 - veličine 32 bita
 - pristupa im se primjenom indeksnog registra **vars**
 - okolina metode**
 - zapisuju se podaci koji se koriste tijekom upravljanja okvirima
 - » kazaljka koja pokazuje na prethodni okvir
 - » kazaljka koja pokazuje na lokalne varijable
 - » kazaljka koja pokazuje na dno stoga operanada
 - » kazaljka koja pokazuje na vrh stoga operanada
 - stog operanada**
 - FIFO stog
 - operandi na stogu su veličine 32 bita

JAVA ByteCode

- **Naredbe Java stroja**
 - koriste stog operanada za dohvat operanada i spremanje rezultata operacije
 1. naredba **iadd** uzme dva cijela broja s vrha stoga
 2. zbroji pročitane brojeve
 3. vrijednost zbroja spremi na vrh stoga
 - **naredbeni skup:**
 - aritmetičke, logičke, pretvorbe vrijednosti obilježja, upravljanje vrijednostima na stogu, upravljanje poljima, upravljanje tijekom izvođenja programa
 - **većina naredbi veličine je jednog okteta**

JAVA ByteCode

- Naredba zbrajanja brojeva ladd
 - veličina naredbe je jedan oktet

Naredba zapisana primjenom mnemonika:

ladd

(Operacijski kôd naredbe)

Naredba zapisana heksadekadski:

61

(Operacijski kôd naredbe)

- **Naredba iload_<n>**
 - pomak n određuje koja se lokalna varijabla dohvaća i stavlja na vrh stoga
 - **veličina naredbe je dva okteta**
 - drugi oktet je pomak n koji se dodaje sadržaju registra vars

Naredba zapisana primjenom mnemonika:

iload_<nn>

Naredba zapisana heksadekadski:

15

nn

(Operacijski kôd naredbe)
(Pomak)

Linearni oblici međukôda

$\underline{x} := Y \text{ op } z \leftarrow$

- Y, Z - adrese operanada
- x - adresa rezultata
- Op - oznaka operacije
- **Linearni oblik**
 - izvršeno je izravnavanje sintaksnog stabla
 - operandi su korisničke varijable, privremene varijable i konstante
 - programski se ostvaruju primjenom kazaljki koje pokazuju na tablicu znakova
 - naredbe koje upravljaju tijekom izvođenja programa koriste simboličke programske oznake

Linearni oblici međukôda

X := Y op Z

Pridruživanje rezultata binarne operacije

X := op Y

Pridruživanje rezultata unarne operacije

X := Y

Preslikavanje

goto L

Naredba bezuvjetnog grananja

if X relop Y goto L

Naredba uvjetnog grananja

call P,n

Poziv potprograma

return Y

Povratak iz potprograma

param X

Definiranje parametara potprograma

X := Y[i], X[i] := Y

Indeksirani dohvati podatka

X := &Y

Pridruživanje adrese

X := *Y, *X := Y

Dohvat podatka primjenom adrese

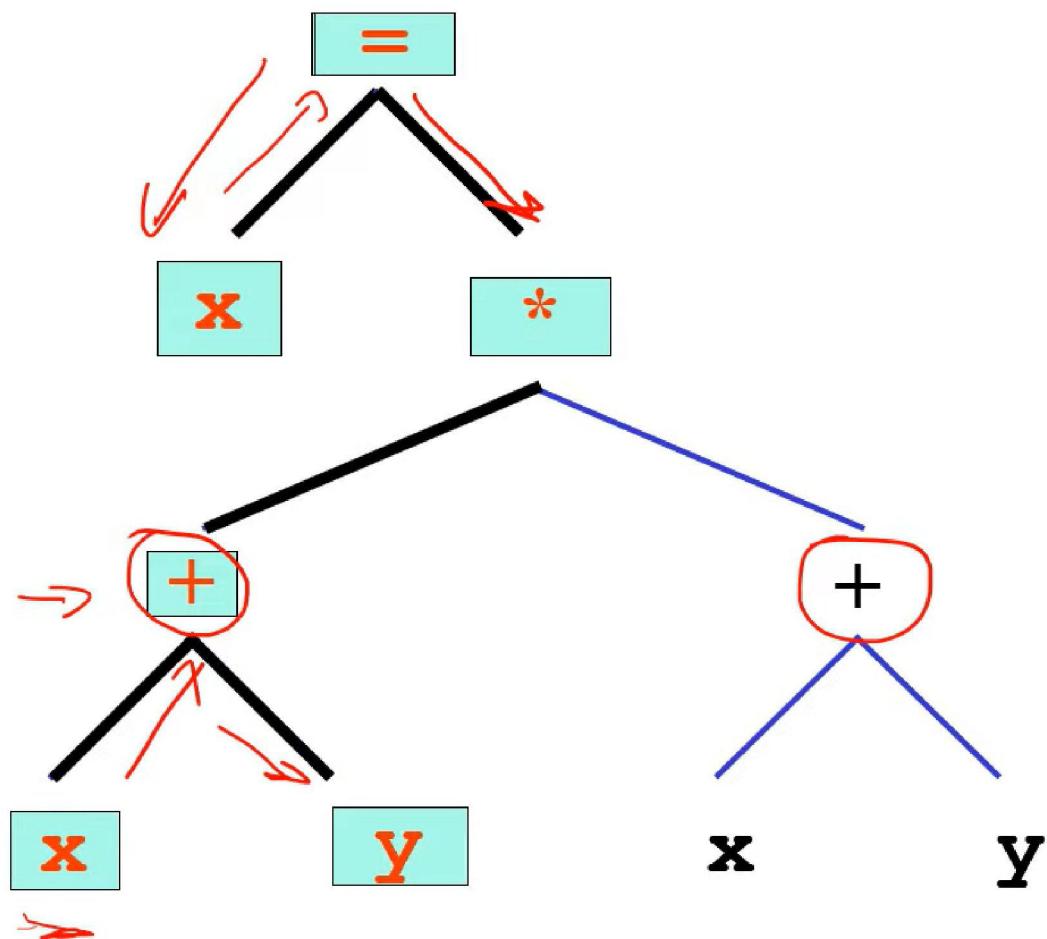
① Izbor skupa operatora

- **Veliki skup operatora**
 - jednostavan prijevod svih naredbi izvornog jezika
- **Mali skup operatora**
 - pojednostavljuje generiranje ciljnog programa
 - generira veliki broj naredbi međukôda
 - otežava optimiranje međukôda

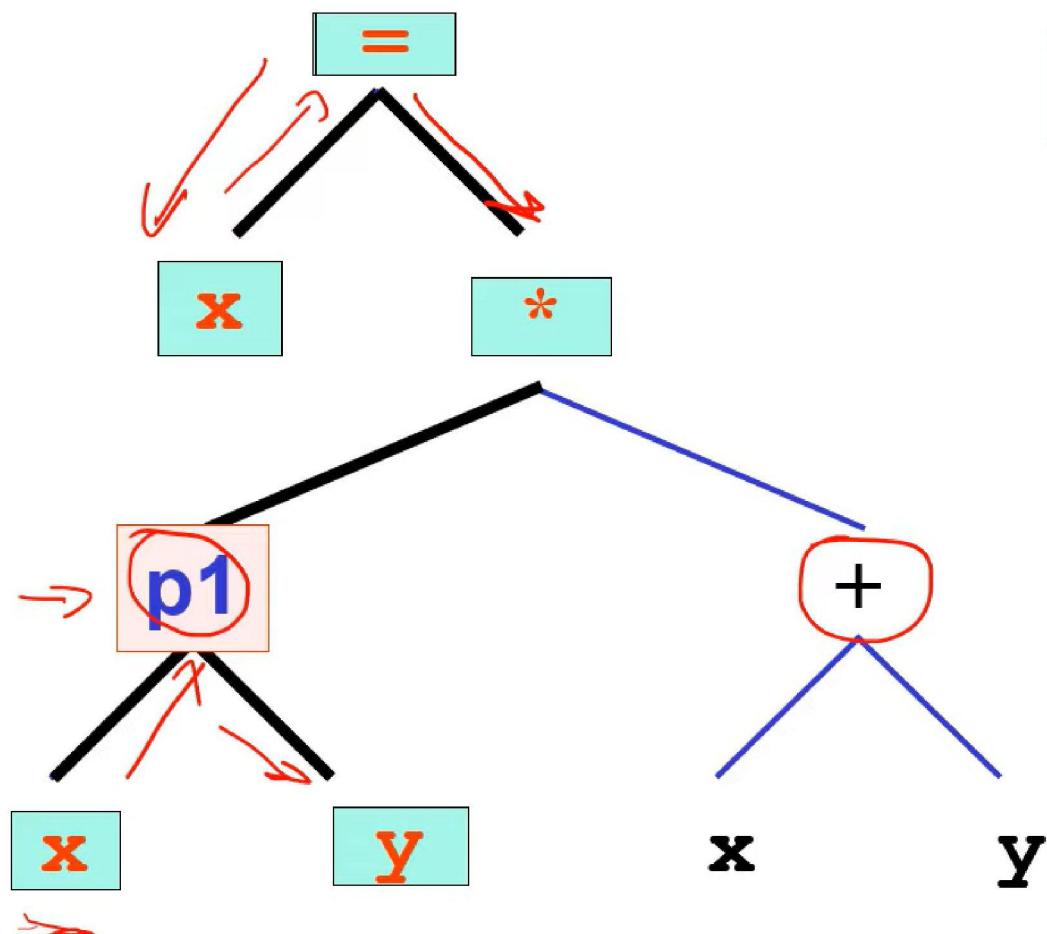
② Pravilni izbor operatora

- pojednostavljuje generiranje i optimiranje međukôda
- pojednostavljuje generiranje ciljnog programa

Linearni oblici međukôda

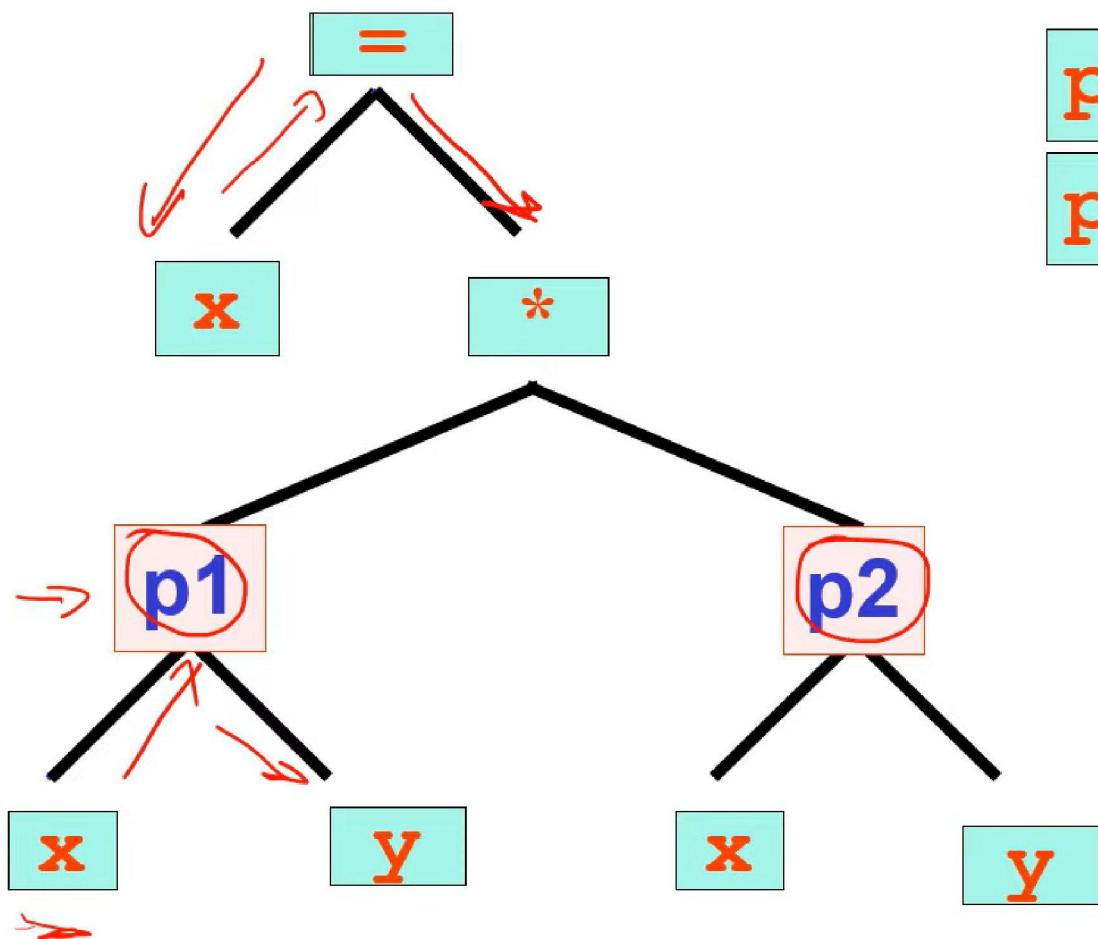


Linearni oblici međukôda



$p1 := x + y$

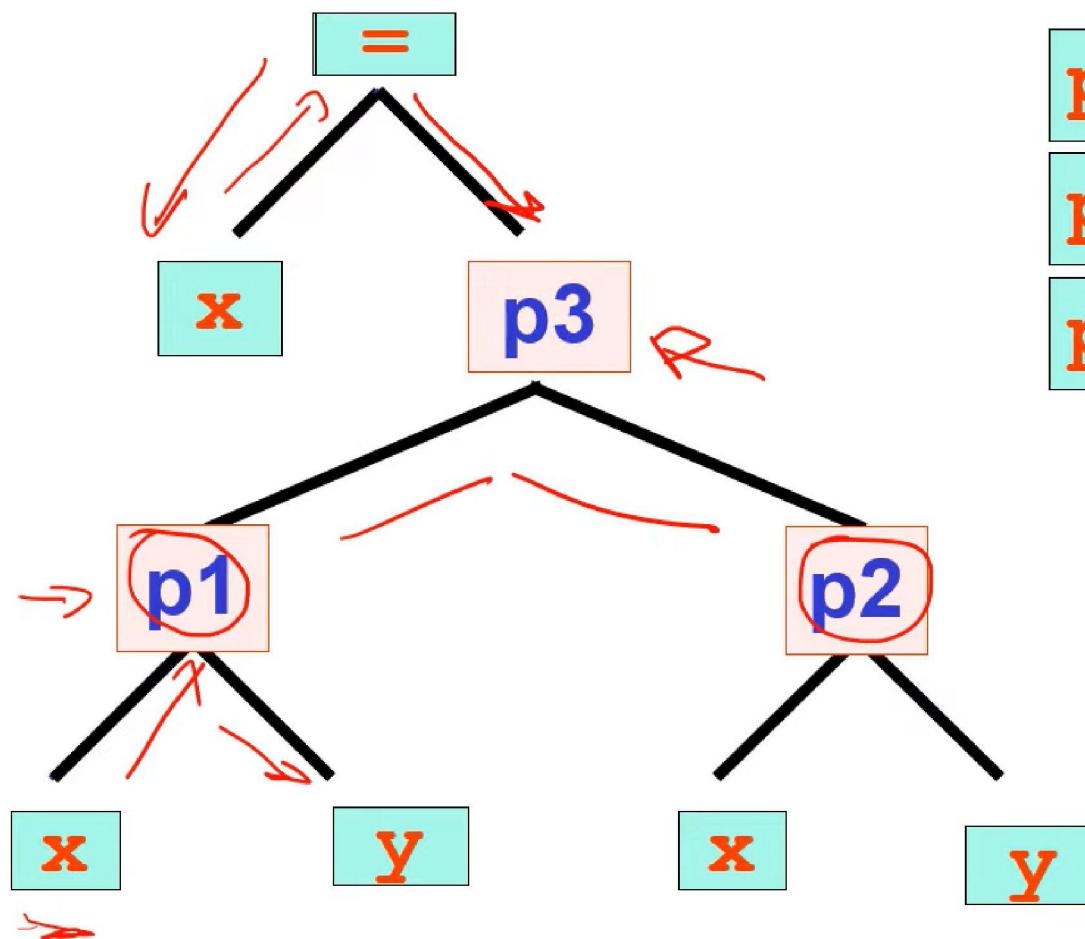
Linearni oblici međukôda



```
p1 := x + y
```

```
p2 := x + y
```

Linearni oblici međukôda

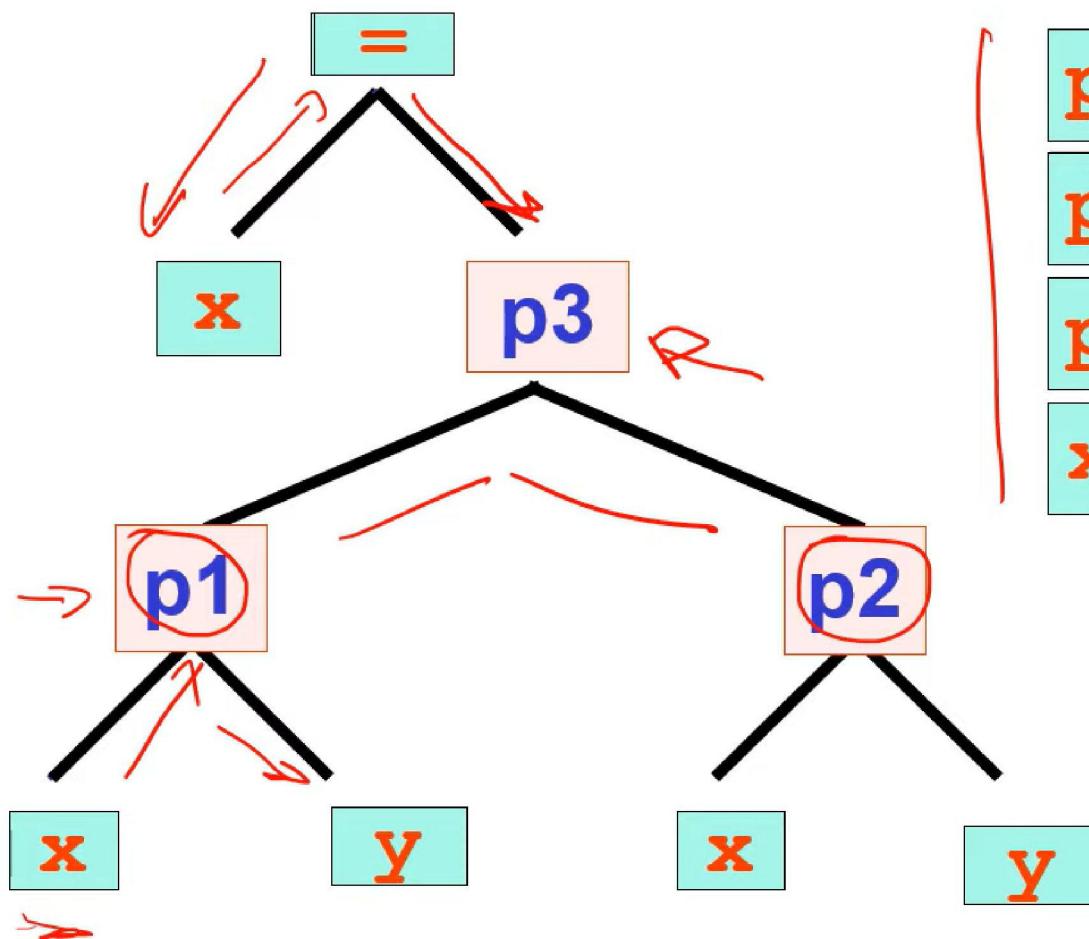


$p1 := x + y$

$p2 := x + y$

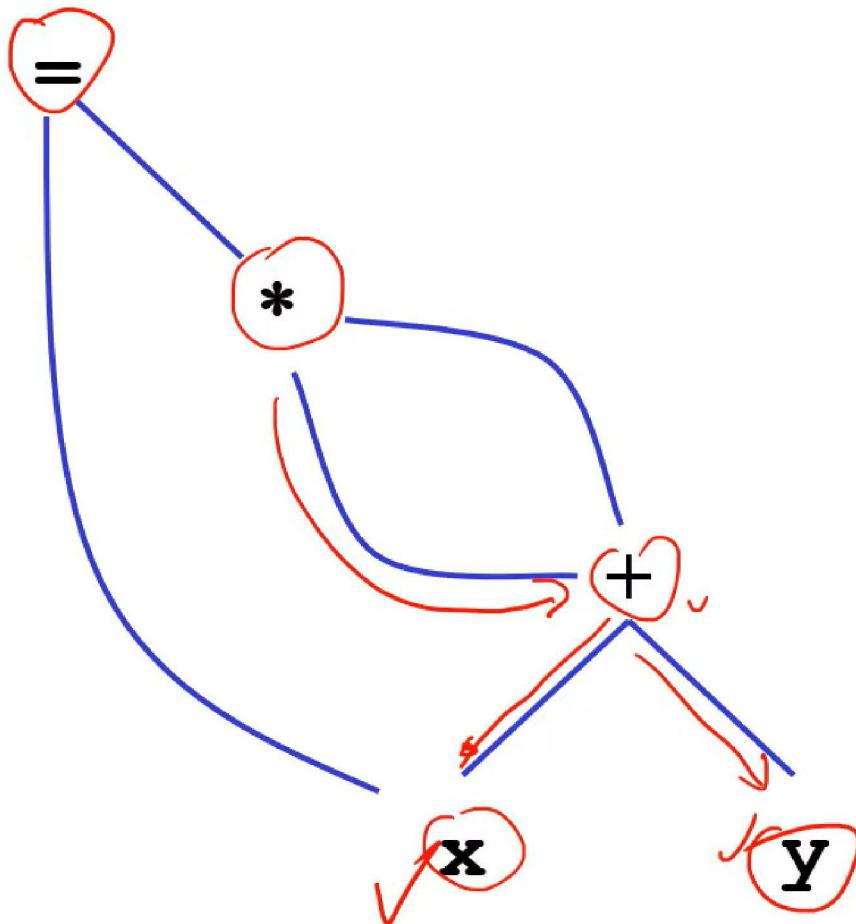
$p3 := p1 * p2$

Linearni oblici međukôda

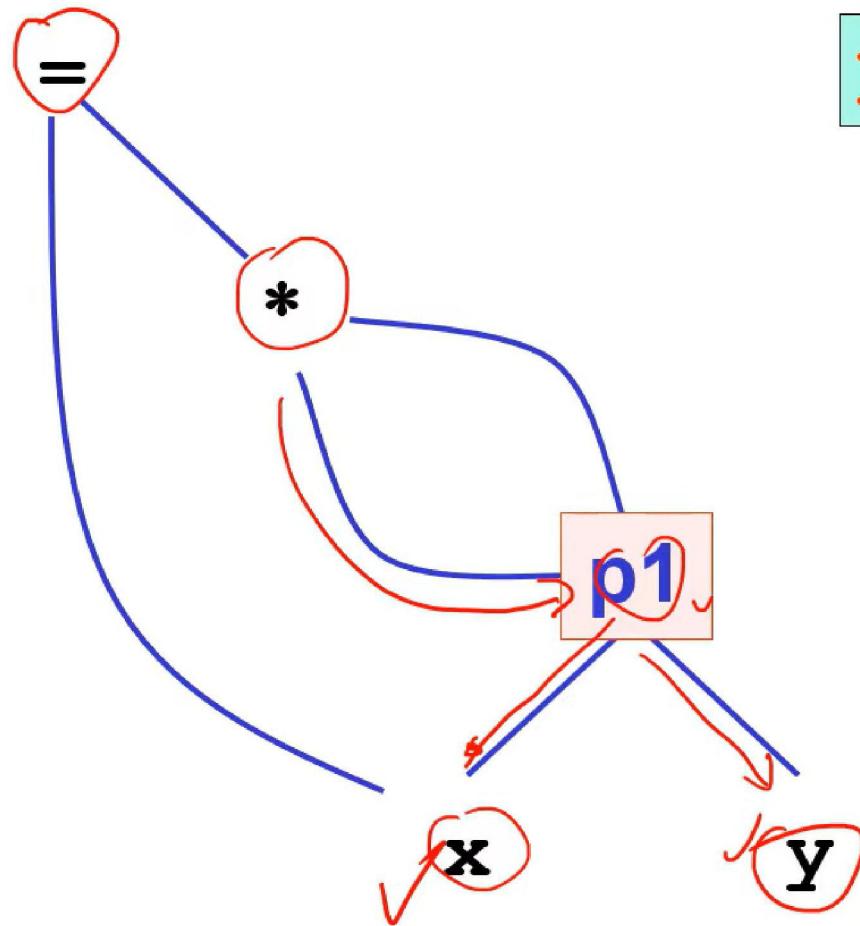


```
p1 := x + y
p2 := x + y
p3 := p1 * p2
x := p3
```

Linearni oblici međukôda

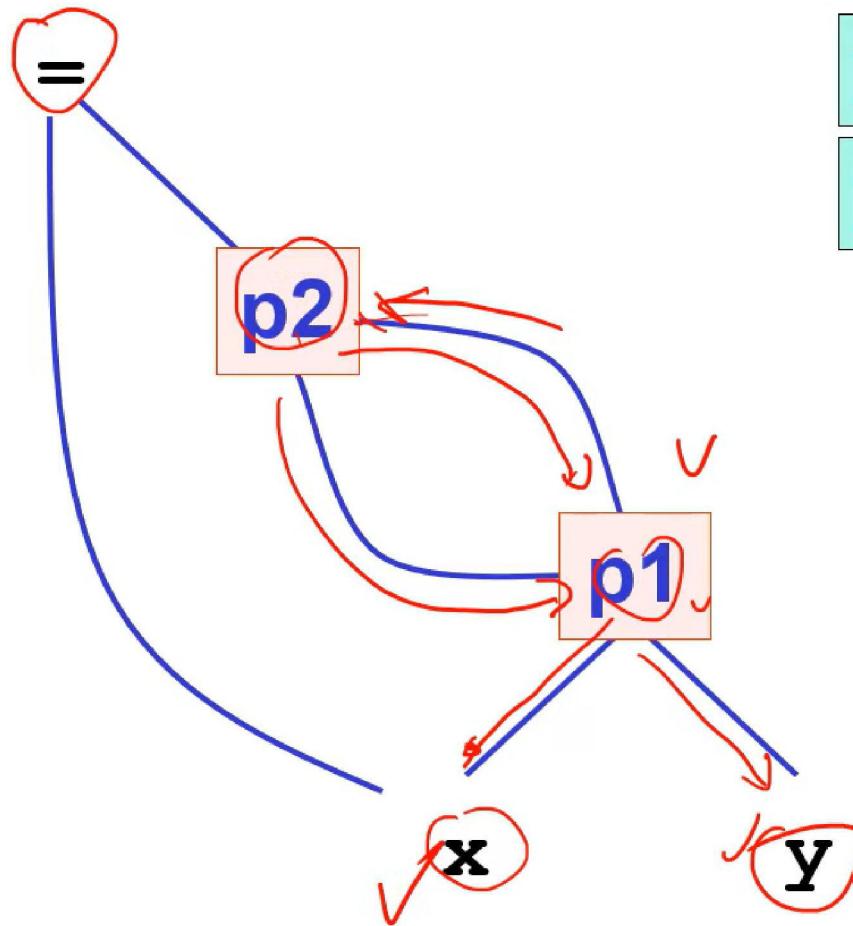


Linearni oblici međukôda



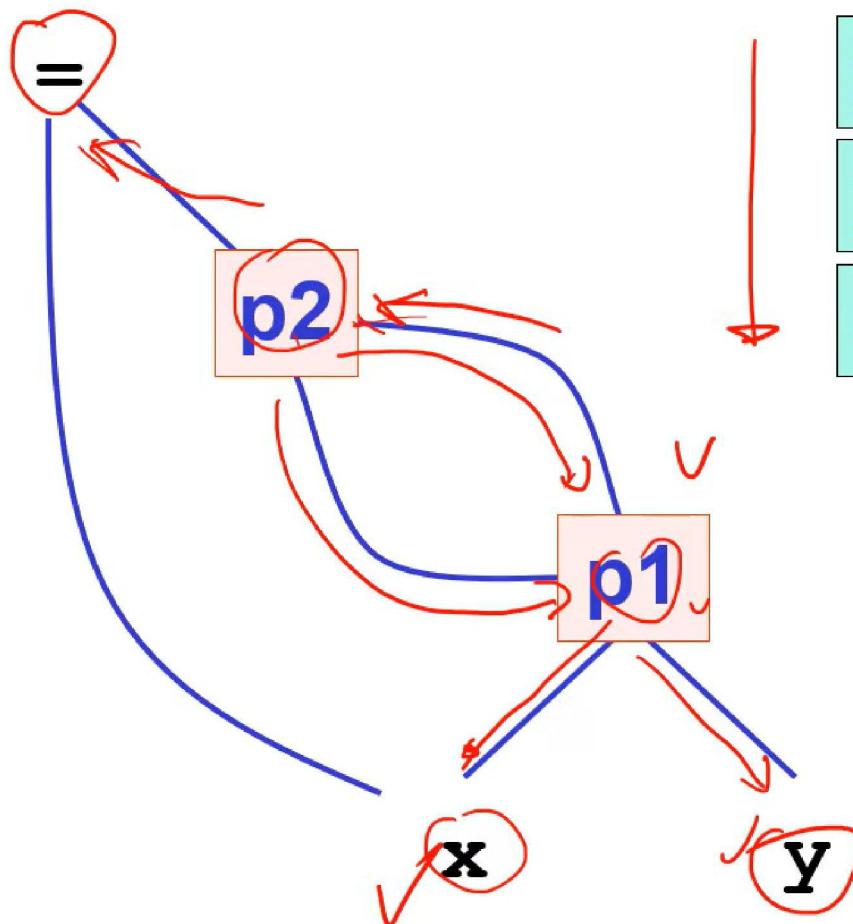
`p1 := x + y`

Linearni oblici međukôda



```
p1 := x + y  
p2 := p1 * p1
```

Linearni oblici međukôda



$p1 := x + y$

$p2 := p1 * p1$

$x := p2$

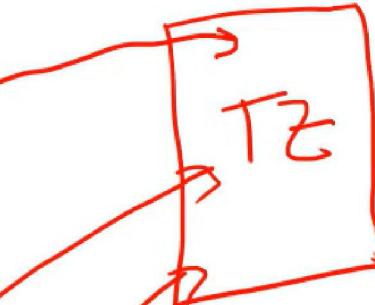
Linearni oblici međukôda



Troadresne naredbe - četvorke

- polje operatora
- dva polja operanda
- polje rezultata

$$\rightarrow x = (x+y) * (x+y)$$



Operator	Prvi operand	Drugi operand	Rezultat
+	x	y	p1
+	x	y	p2
*	p1	p2	p3
=	p3		x

Linearni oblici međukôda

- Troadresne naredbe – trojke
 - polje operatora
 - dva polja operanda

$$\rightarrow x = (x+y) * (x+y)$$

	Operator	Prvi operand	Drugi operand
(1)	+	x	y
(2)	+	x	y
(3)	*	(1)	(2)
(4)	=	(3)	x



Linearni oblici međukôda

6 • Prednost trojki

- učinkovitije korištenje memorijskog prostora
- pogodne su za generiranje ciljnog programa

— sadrže izravni podatak o načinu prenošenja rezultata jedne troadresne naredbe u drugu

• Nedostatak trojki

• nemogućnost učinkovitog mijenjanja redoslijeda izvođenja trojki

- posebice tijekom optimiranja
- promijeni se redoslijed izvođenja trojki
 - mijenjaju se oznaće u poljima operanada

Linearni oblici međukôda

- **Troadresne naredbe - neizravne trojke**
 - dodatni vektor izvođenja
 - učinkovita promjena redoslijeda izvođenja trojki

$$\mathbf{x} = (\mathbf{x}+\mathbf{y}) * (\mathbf{x}+\mathbf{y})$$

Vektor
izvođenja

Trojka	Operator	Prvi operand	Drugi operand
(1) (17)	(17)	x	y
(2) (18)	(18)	x	y
(3) (19)	(19)	(17)	(18)
(4) (20)	(20)	(19)	x

Linearni oblici međukôda

- **Troadresne naredbe - neizravne zdržene trojke**
 - učinkovitije se koristi memorijski prostor

Vektor
izvođenja

$$\underline{x} + \underline{y}$$

$$x = (x+y) * (x+y)$$

Trojka	Operator	Prvi operand	Drugi operand
(1) (17)	(17)	x	y
(2) (17)	(18)	(17)	(17)
(3) (18)	(19)	(18)	x
(4) (19)			

Linearni oblici međukôda

- **Troadresne naredbe - neizravne združene trojke**

- učinkovitije se koristi memorijski prostor

- Ako se ne mijenja vrijednost operanada,
- onda je moguće izostaviti oznaku trojke u vektoru izvođenja

Vektor
izvođenja

$$\mathbf{x} = (\mathbf{x+y}) * (\mathbf{x+y})$$

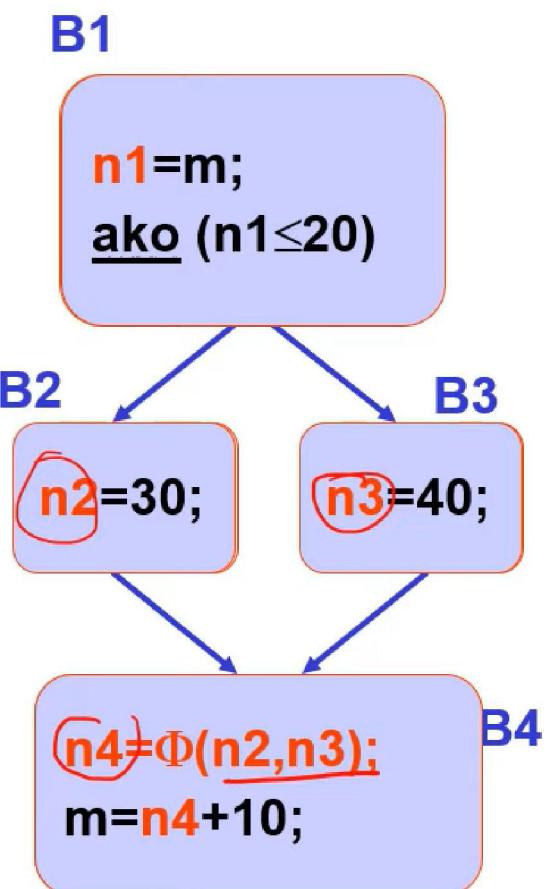
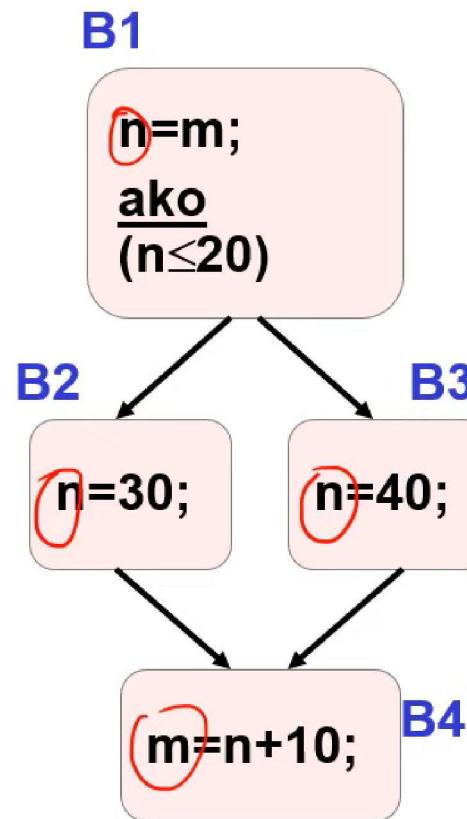
Trojka	Operator	Prvi operand	Drugi operand
(1) (17)	+	x	y
(18)	*	(17)	(17)
(3) (18)	=	(18)	x
(4) (19)			

Posebni oblici međukôda

- Statičko jednostruko pridruživanje
 - Prilagođeno postupcima optimiranja

```

n = m;
ako ( $n \leq 20$ )
  n = 30
inache
    n = 40;
    m = n+10;
  
```



Posebni oblici međukôda

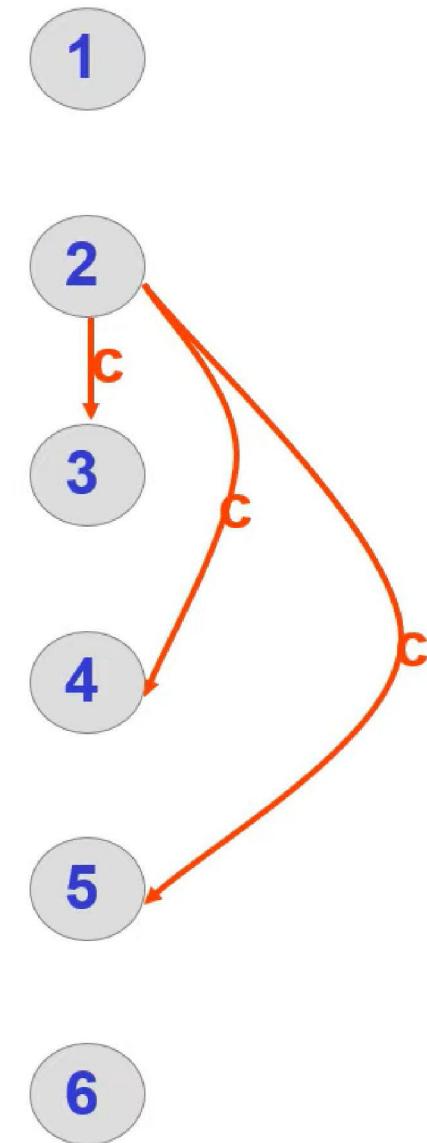
Graf zavisnosti

Graf zavisnosti upravljačkog tijeka

Graf zavisnosti podataka

Posebni oblici međukôda

```
1)      i = j + k;  
2)      ako ( i > 37 )  
3)          skoči L1  
        inac̄e  
        {  
4)          n = j * m;  
5)          m = n + 13;  
        }  
6) L1:   n = m / 23;
```



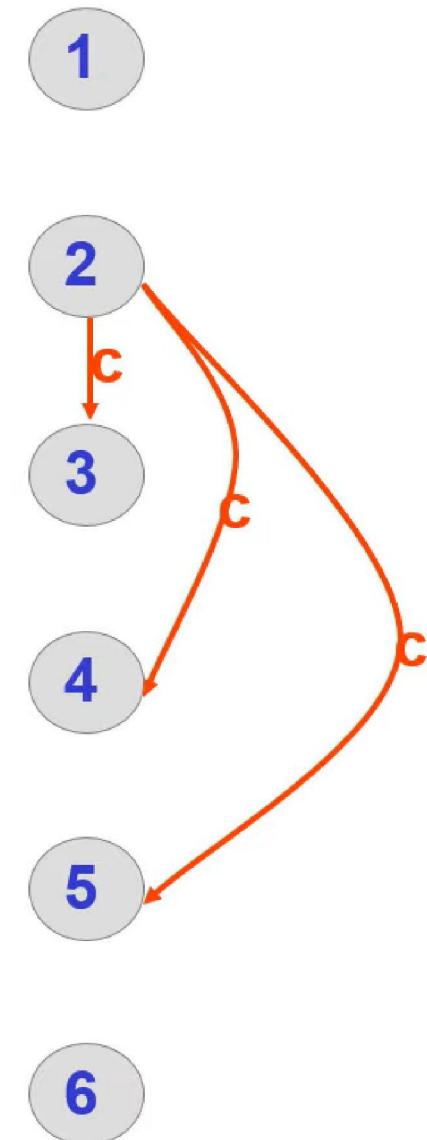
Posebni oblici međukôda

Naredba N1 mijenja vrijednost podatka

Naredba N2 koristi

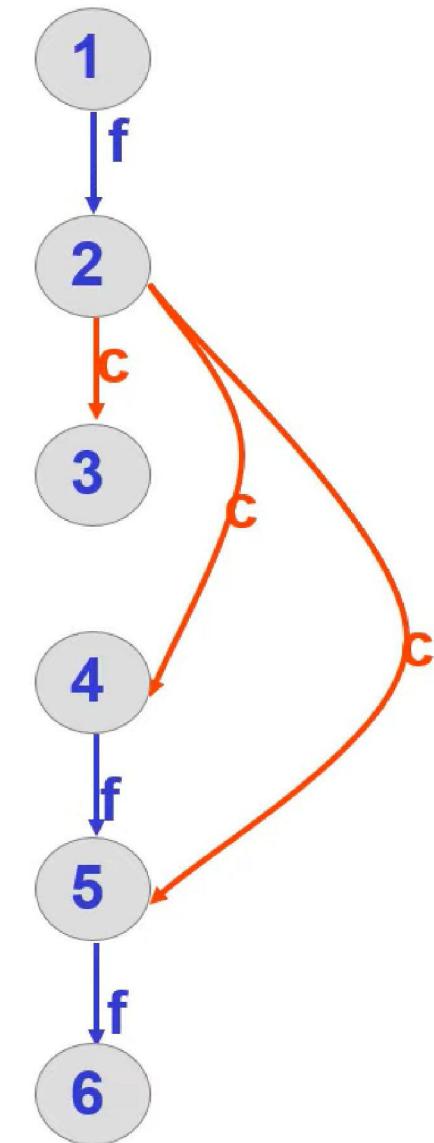
N1 i N2 \Rightarrow unaprijedno zavisne (f)

- 1) $i = j + k;$
- 2) ako ($i > 37$)
- 3) skoči L1
inac̄e
{
- 4) $n = j * m;$
- 5) $m = n + 13;$
- 6) }
6) L1: $n = m / 23;$



Posebni oblici međukôda

```
1)      i = j + k;  
2)      ako ( i > 37 )  
3)          skoči L1  
           inache  
           {  
4)          n = j * m;  
5)          m = n + 13;  
           }  
6) L1:   n = m / 23;
```



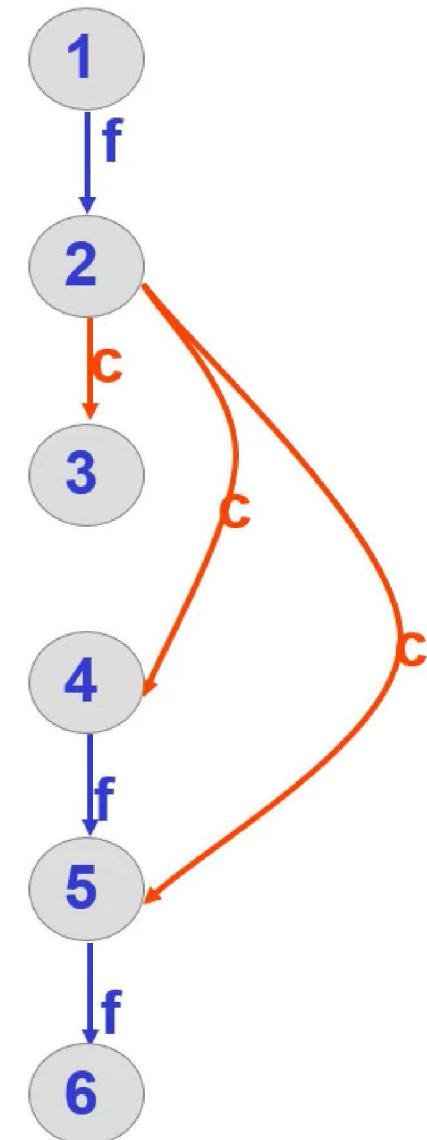
Posebni oblici međukôda

Naredba N2 mijenja vrijednost podatka

Naredba N1 koristi

N1 i N2 \Rightarrow unazadno zavisne (a)

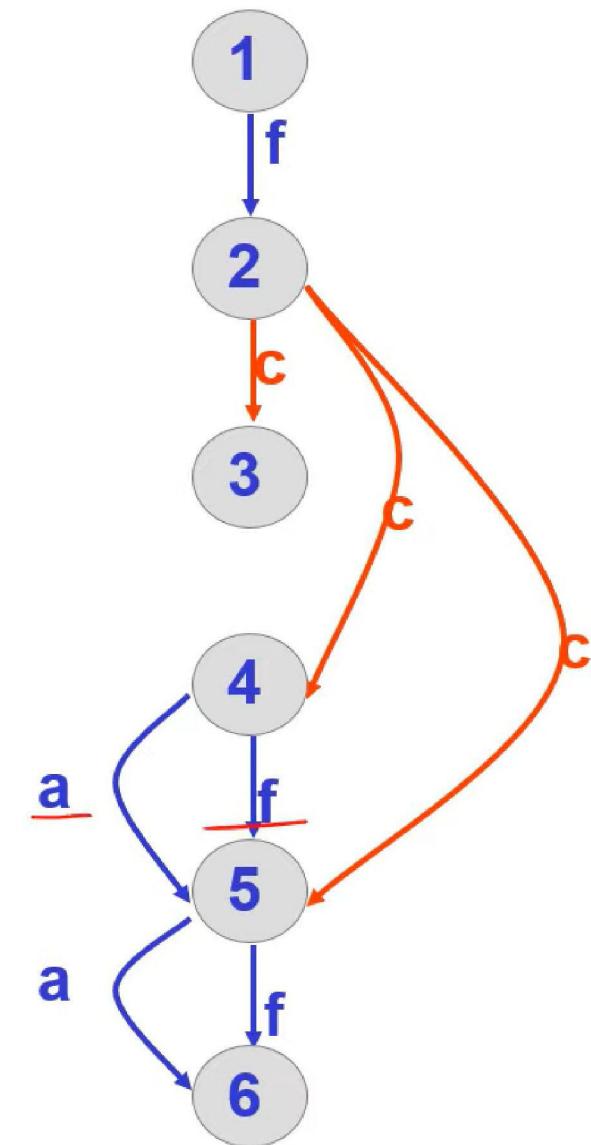
- 1) $i = j + k;$
- 2) ako ($i > 37$)
- 3) skoči L1
inac̄e
{
- 4) $n = j * m;$
- 5) $m = n + 13;$
- 6) }
6) L1: $n = m / 23;$



Posebni oblici međukôda

```

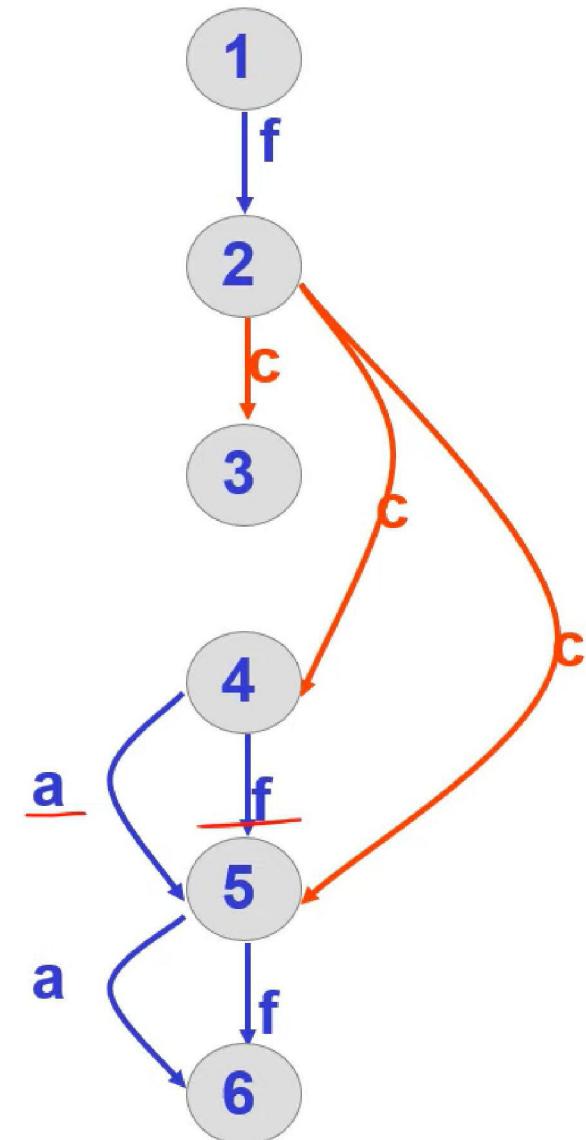
1)   i = j + k;
2)   ako (i > 37 )
3)       skoči L1
        inache
        {
4)           n = j * (m);
5)           m = n + 13;
        }
6) L1:  n = m / 23;
    
```



Posebni oblici međukôda

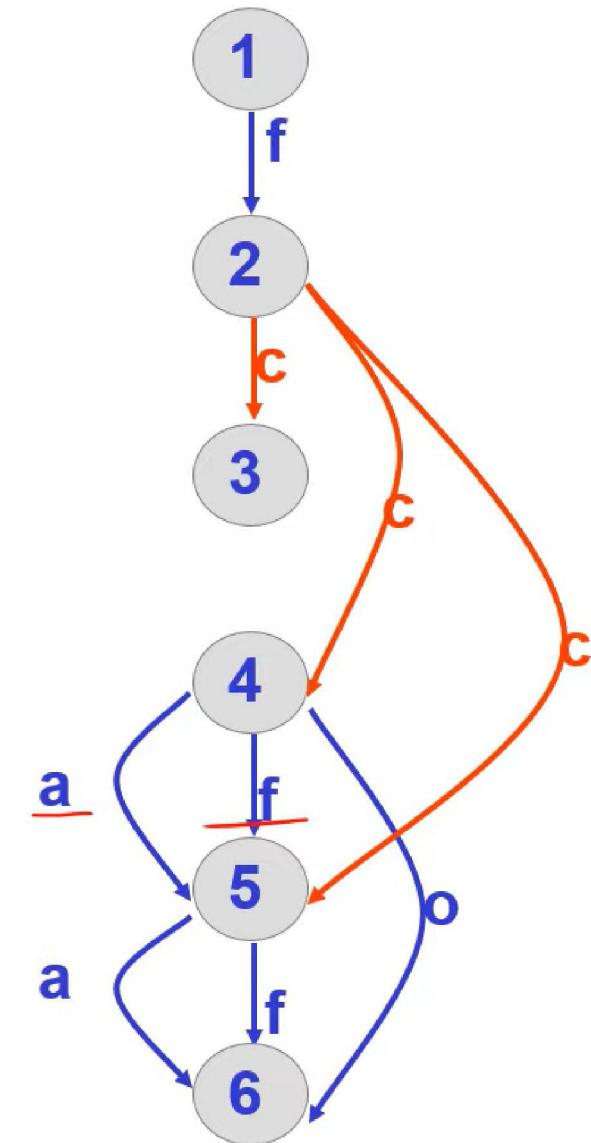
Naredbe N1 i N2 obje mijenjaju
vrijednost podatka
N1 i N2 \Rightarrow zavisnost odredišta (o)

- 1) $i = j + k;$
- 2) ako ($i > 37$)
- 3) skoči L1
- inache
- {
- 4) $n = j * m;$
- 5) $m = n + 13;$
- }
- 6) L1: $n = m / 23;$



Posebni oblici međukôda

- 1) $i = j + k;$
- 2) ako ($i > 37$)
- 3) skoči L1
- 4)
- 5) $n = j * m;$
 $m = n + 13;$
- 6) L1: $n = m / 23;$



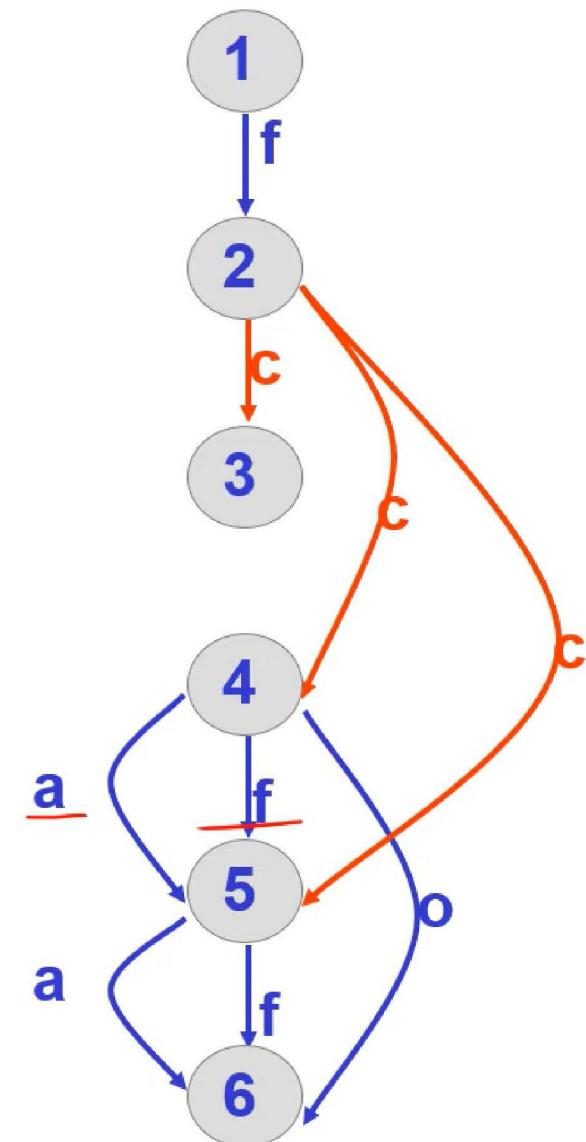
Posebni oblici međukôda

Naredbe N1 i N2 obje koriste

vrijednost podatka

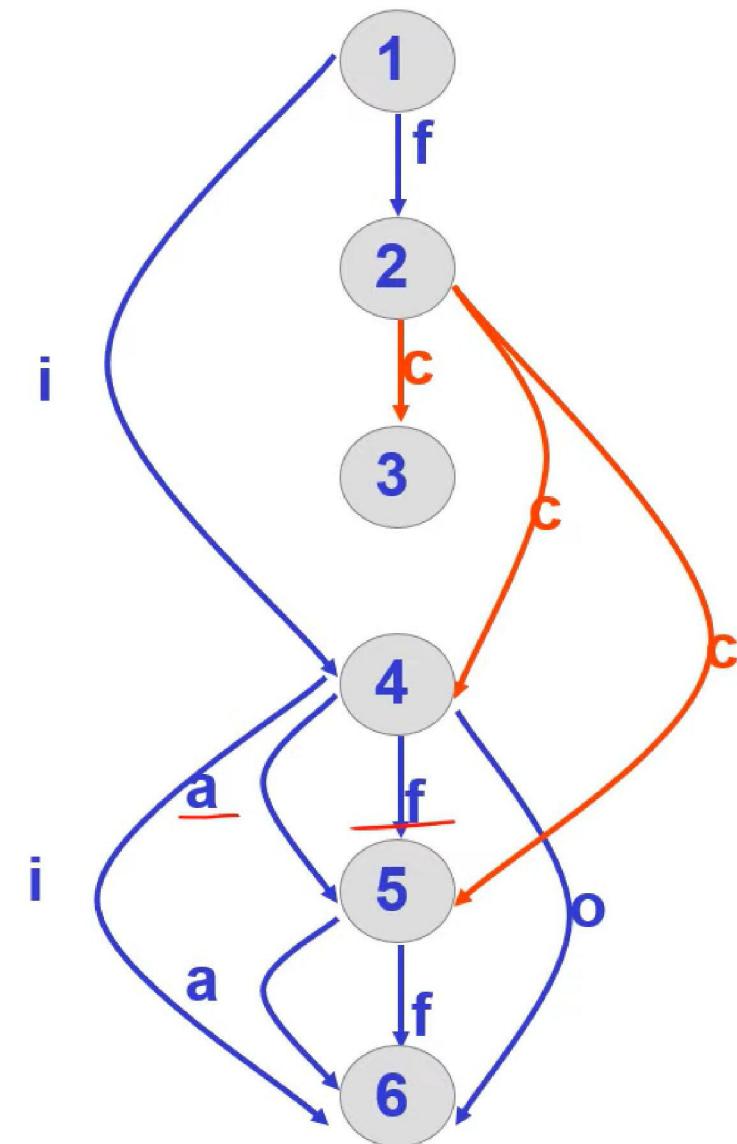
N1 i N2 \Rightarrow zavisnost izvorišta (i)

- 1) $i = j + k;$
- 2) ako ($i > 37$)
- 3) skoči L1
- inache
- {
- 4) $n = j * m;$
- 5) $m = n + 13;$
- }
- 6) L1: $n = m / 23;$



Posebni oblici međukôda

- 1) $i = j + k;$
 - 2) ako ($i > 37$)
 - 3) skoči L1
 - 4)
 - 5) $n = j * m;$
 $m = n + 13;$
 - 6) L1: $n = m / 23;$
- inache
{

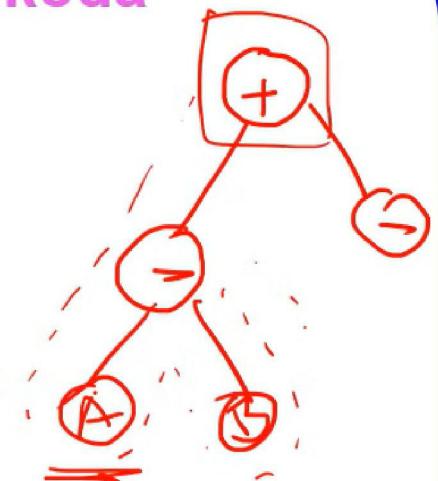


Sintaksom upravljano generiranje međukôda

- Primjena atributne prijevodne gramatike
 - Generiranje sažetog sintaksnog stabla
 - Generiranje troadresnih naredbi

Sintaksom upravljano generiranje međukôda

- Primjena atributne prijevodne gramatike
 - Generiranje sažetog sintaksnog stabla



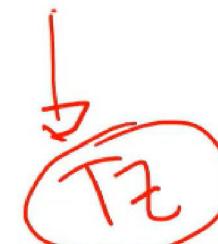
$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3} \quad \{ k_1 = \text{StvoriČvor}("=", \text{StvoriList}(k_2), k_3); \}$

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3} \quad \{ k_1 = \text{StvoriČvor}("+", k_2, k_3); \}$

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3} \quad \{ k_1 = \text{StvoriČvor}("*", k_2, k_3); \}$

$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2}) \quad \{ k_1 = k_2; \}$

$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2} \quad \{ k_1 = \text{StvoriList}(k_2); \}$



$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k_2), k_3);$ }

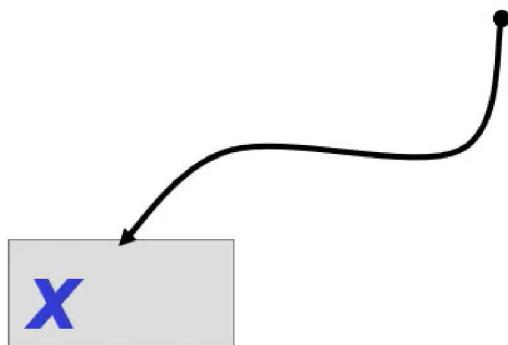
$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“+”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3}$ { $k_1 = \text{StvorČvor}(“*”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2})$ { $k_1 = k_2;$ }

$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2}$ { $k_1 = \text{StvoriList}(k_2);$ }

$$\Rightarrow x = (\underline{x} + y) * (\underline{x} + y)$$



$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k_2), k_3);$ }

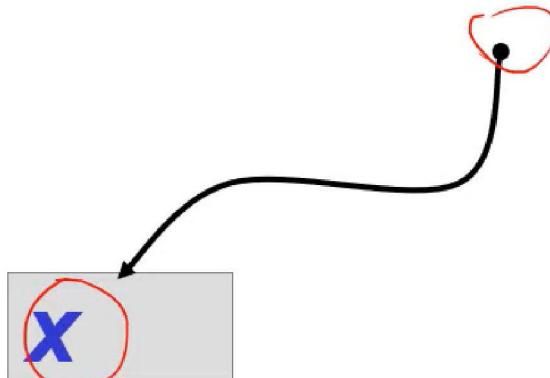
$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“+”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3}$ { $k_1 = \text{StvorČvor}(“*”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2})$ { $k_1 = k_2;$ }

$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2}$ { $k_1 = \text{StvoriList}(k_2);$ }

$$\Rightarrow x = (\underline{E} + y) * (x + y)$$



$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k_2), k_3);$ }

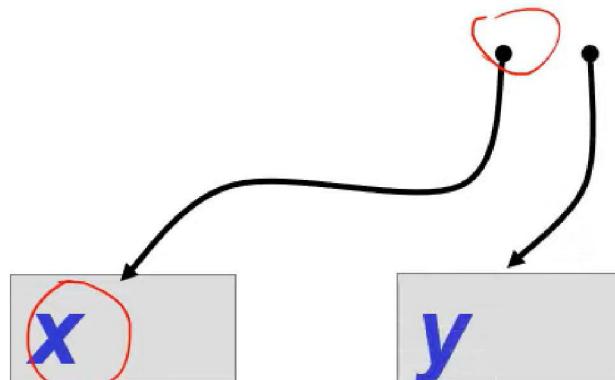
$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“+”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3}$ { $k_1 = \text{StvorČvor}(“*”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2})$ { $k_1 = k_2;$ }

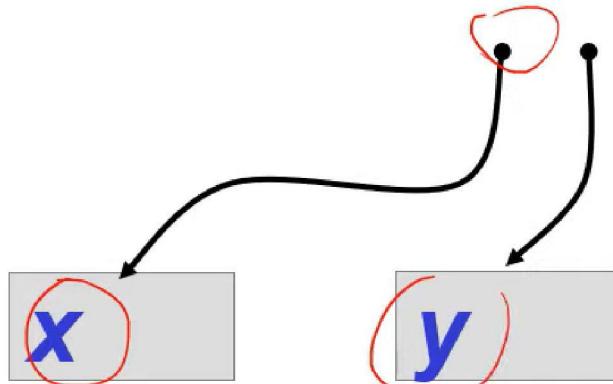
$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2}$ { $k_1 = \text{StvoriList}(k_2);$ }

$$\Rightarrow x = (\underline{E} + \underline{y}) * (x + y)$$



$\langle S \rangle_{k1} \rightarrow \text{IDN}_{k2} = \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k2), k3);$ }
 $\langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} + \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}(“+”, k2, k3);$ }
 $\langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} * \langle E \rangle_{k3}$ { $k1 = \text{StvorČvor}(“*”, k2, k3);$ }
 $\langle E \rangle_{k1} \rightarrow (\langle E \rangle_{k2})$ { $k1 = k2;$ }
 $\langle E \rangle_{k1} \rightarrow \text{IDN}_{k2}$ { $k1 = \text{StvoriList}(k2);$ }

$\Rightarrow x = (\underline{\underline{E}} + \underline{\underline{E}}) * (x + y)$



$\langle S \rangle_{k1} \rightarrow \text{IDN}_{k2} = \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}("=", \text{StvoriList}(k2), k3);$ }

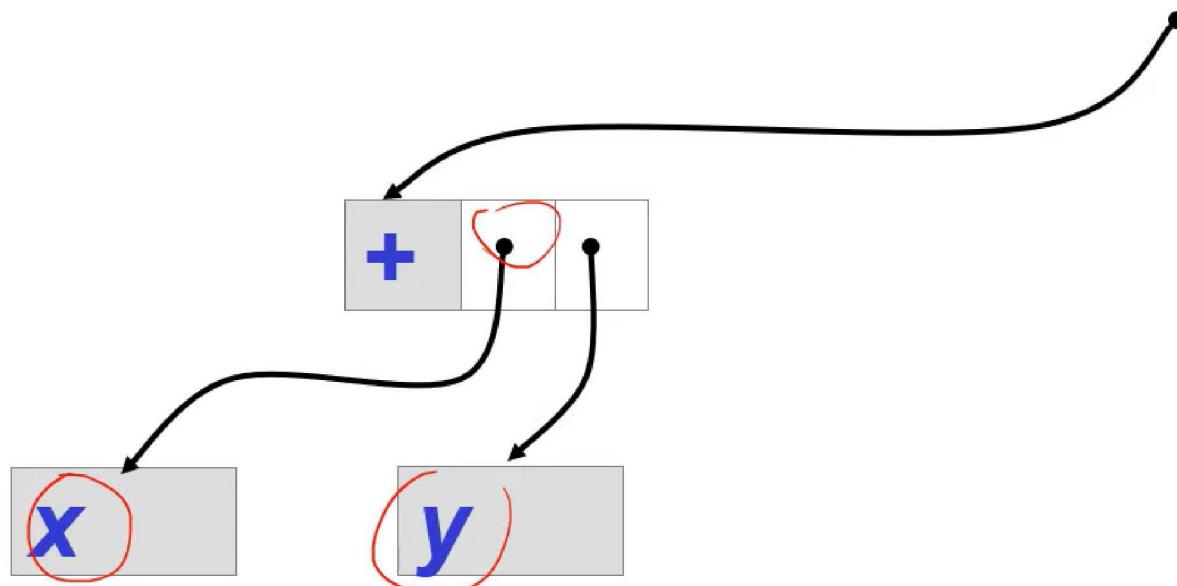
$\Rightarrow \langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} + \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}("+" \circled, k2, k3);$ }

$\langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} * \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}("*", k2, k3);$ }

$\langle E \rangle_{k1} \rightarrow (\langle E \rangle_{k2})$ { $k1 = k2;$ }

$\langle E \rangle_{k1} \rightarrow \text{IDN}_{k2}$ { $k1 = \text{StvoriList}(k2);$ }

$$\Rightarrow x = (\underline{\underline{E + E}}) * (\underline{x + y})$$



$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k_2), k_3);$ }

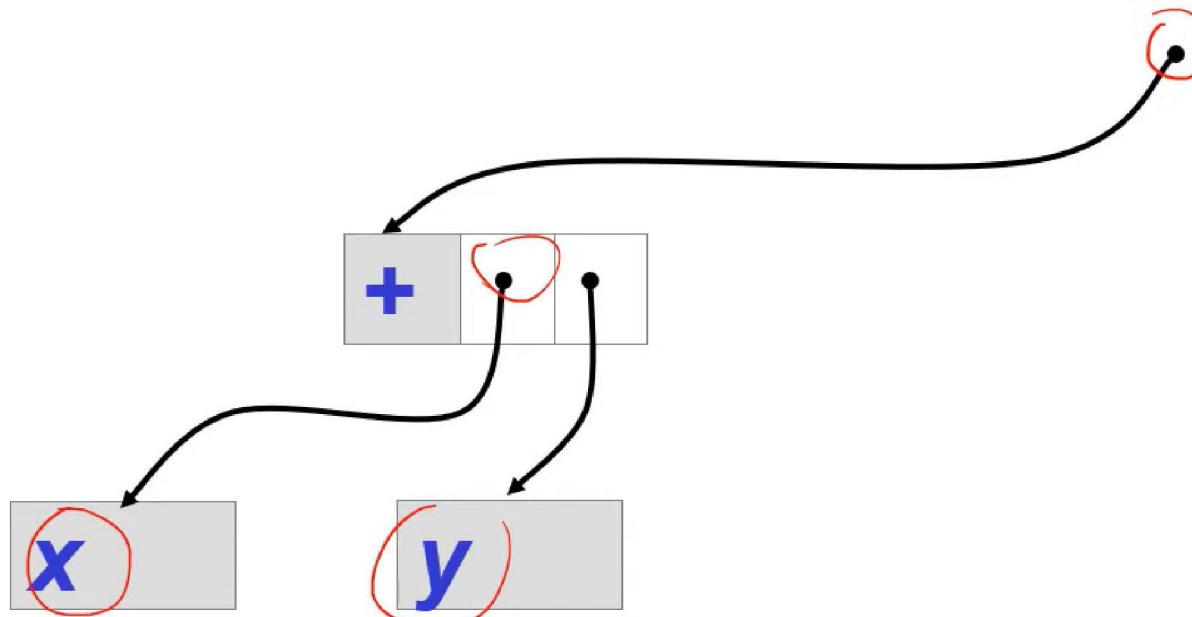
$\Rightarrow \langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“+”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3}$ { $k_1 = \text{StvorČvor}(“*”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2})$ { $k_1 = k_2;$ }

$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2}$ { $k_1 = \text{StvoriList}(k_2);$ }

$$\Rightarrow x = (\underline{\underline{E}}) * (x + y)$$



$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k_2), k_3);$ }

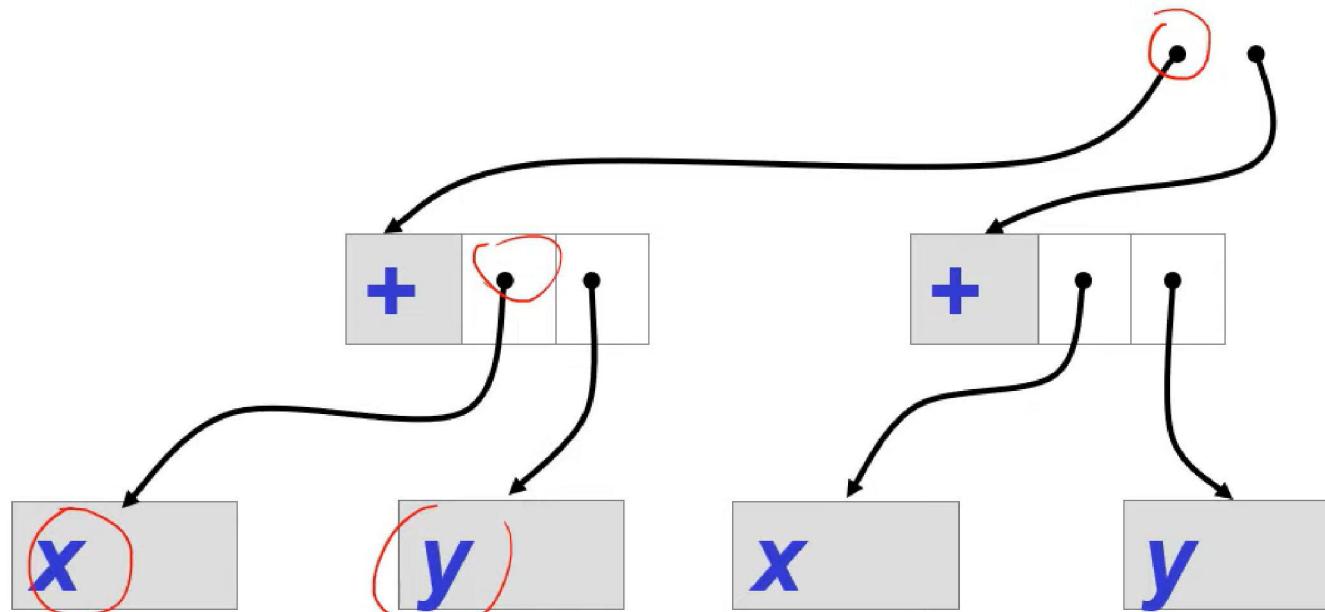
$\Rightarrow \langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“+”, k_2, k_3);$ }

$\langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3}$ { $k_1 = \text{StvoriČvor}(“*”, k_2, k_3);$ }

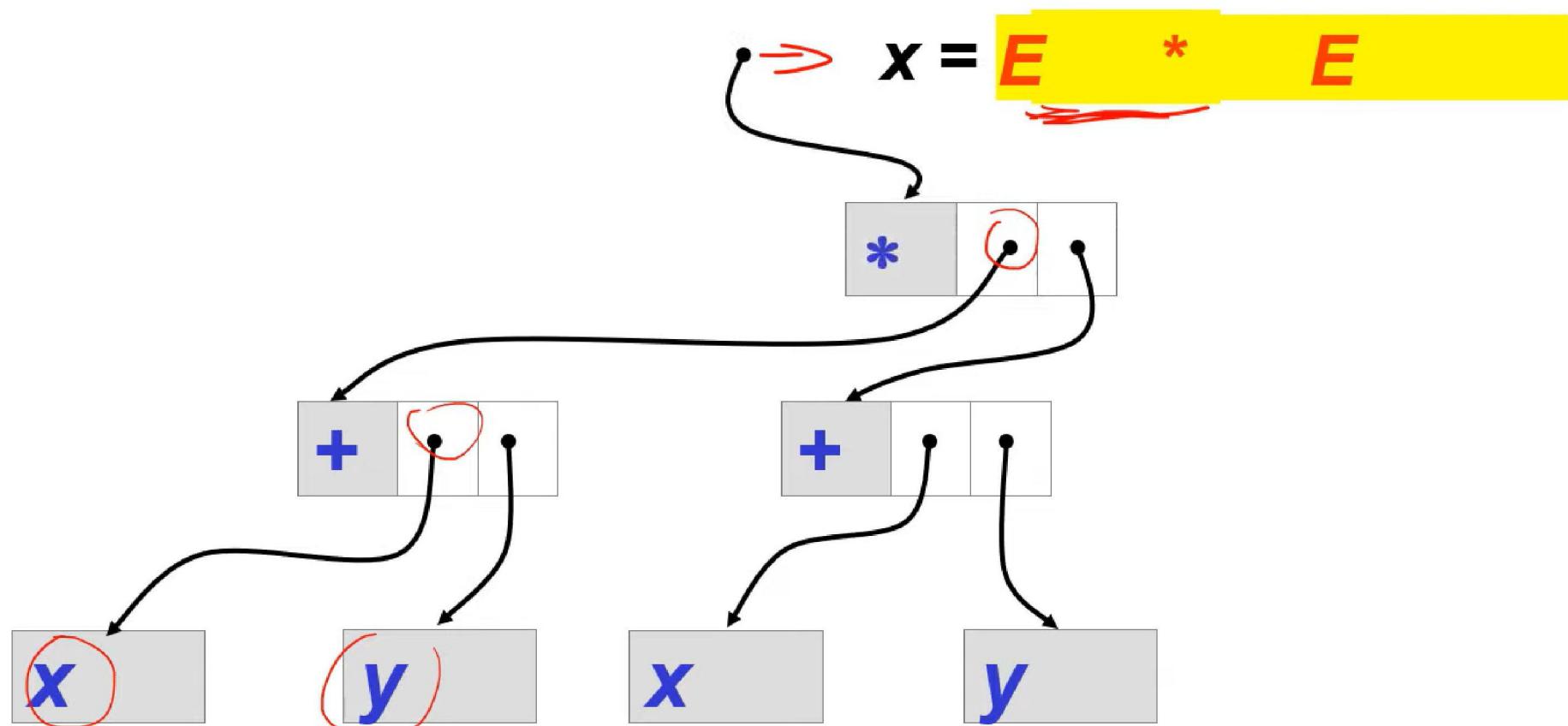
$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2})$ { $k_1 = k_2;$ }

$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2}$ { $k_1 = \text{StvoriList}(k_2);$ }

$$\Rightarrow x = E * E$$



$\langle S \rangle_{k1} \rightarrow \text{IDN}_{k2} = \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}("=", \text{StvoriList}(k2), k3);$ }
 $\Rightarrow \langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} + \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}("+", k2, k3);$ }
 $\Rightarrow \langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} * \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}("*", k2, k3);$ }
 $\langle E \rangle_{k1} \rightarrow (\langle E \rangle_{k2})$ { $k1 = k2;$ }
 $\langle E \rangle_{k1} \rightarrow \text{IDN}_{k2}$ { $k1 = \text{StvoriList}(k2);$ }



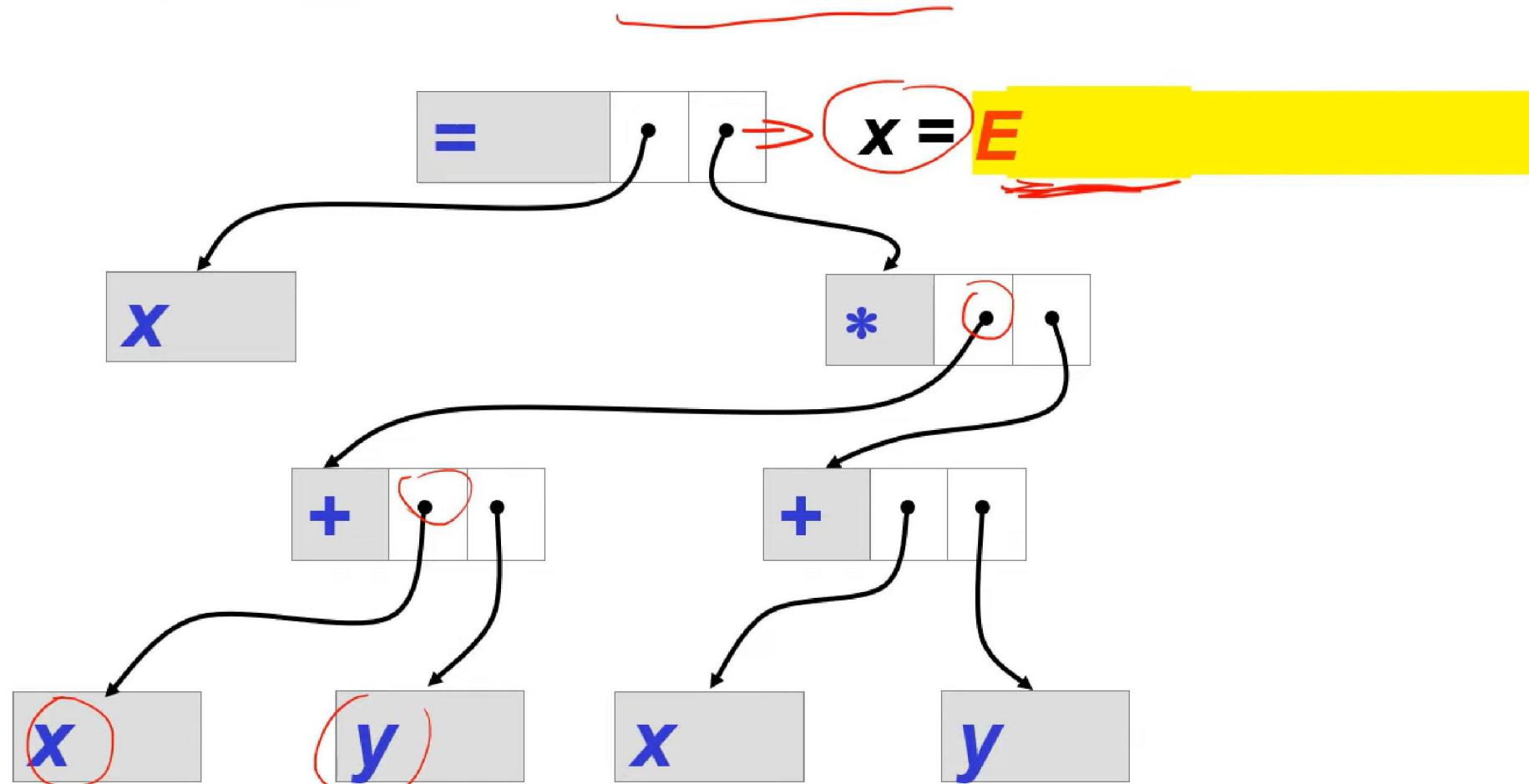
$\langle S \rangle_{k_1} \rightarrow \text{IDN}_{k_2} = \langle E \rangle_{k_3} \quad \{ k_1 = \text{StvoriČvor}(“=”, StvoriList(k_2), k_3); \}$

$\Rightarrow \langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} + \langle E \rangle_{k_3} \quad \{ k_1 = \text{StvoriČvor}(“+”, k_2, k_3); \}$

$\Rightarrow \langle E \rangle_{k_1} \rightarrow \langle E \rangle_{k_2} * \langle E \rangle_{k_3} \quad \{ k_1 = \text{StvoriČvor}(“*”, k_2, k_3); \}$

$\langle E \rangle_{k_1} \rightarrow (\langle E \rangle_{k_2}) \quad \{ k_1 = k_2; \}$

$\langle E \rangle_{k_1} \rightarrow \text{IDN}_{k_2} \quad \{ k_1 = \text{StvoriList}(k_2); \}$



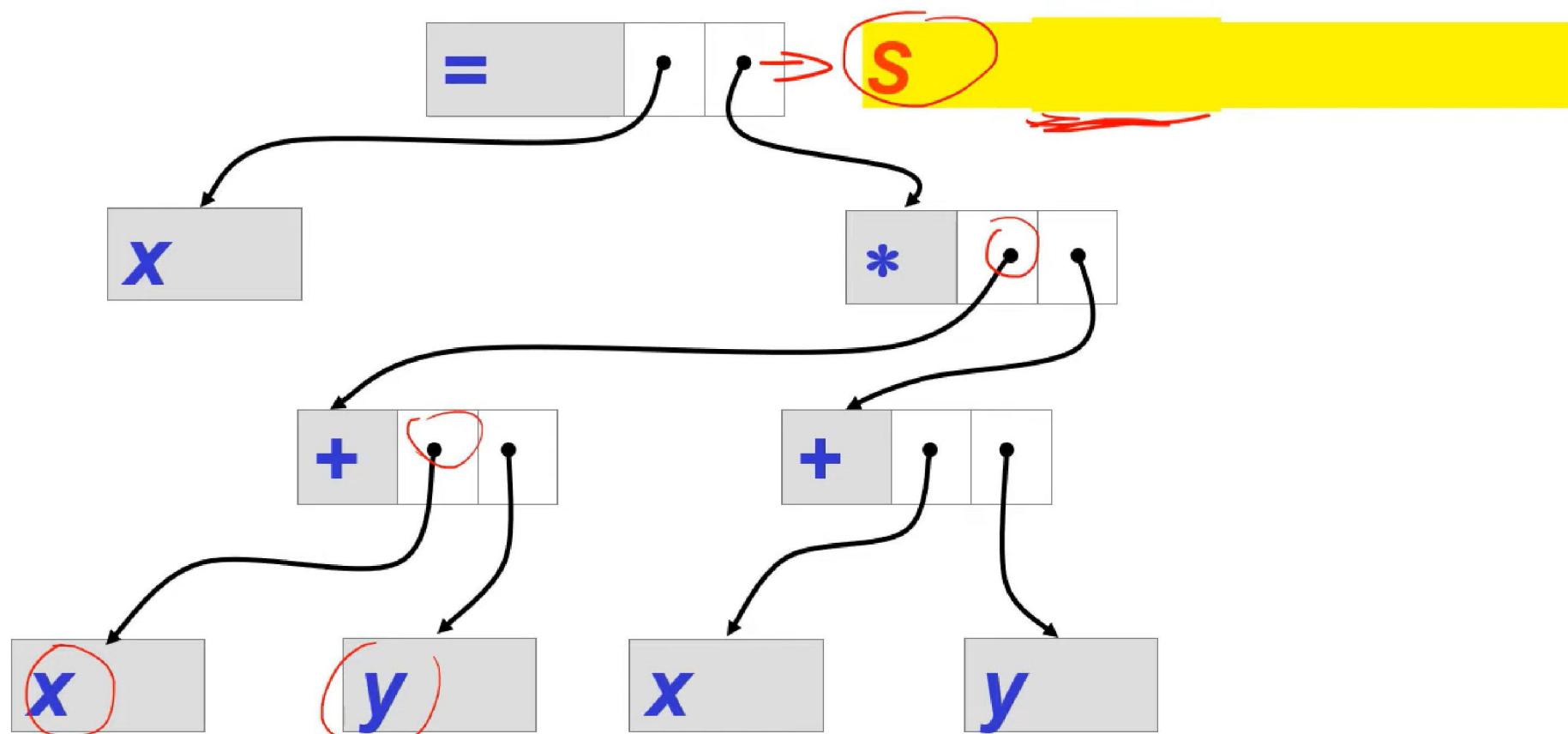
$\langle S \rangle_{k1} \rightarrow \text{IDN}_{k2} = \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}(“=”, \text{StvoriList}(k2), k3);$ }

$\Rightarrow \langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} + \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}(“+”, k2, k3);$ }

$\Rightarrow \langle E \rangle_{k1} \rightarrow \langle E \rangle_{k2} * \langle E \rangle_{k3}$ { $k1 = \text{StvoriČvor}(“*”, k2, k3);$ }

$\langle E \rangle_{k1} \rightarrow (\langle E \rangle_{k2})$ { $k1 = k2;$ }

$\langle E \rangle_{k1} \rightarrow \text{IDN}_{k2}$ { $k1 = \text{StvoriList}(k2);$ }



Sintaksom upravljano generiranje međukôda

- Primjena atributne prijevodne gramatike
 - Generiranje troadresnih naredbi

$\langle E \rangle_{\text{Ime1}, \text{Kod1}} \rightarrow - \langle E \rangle_{\text{Ime2}, \text{Kod2}}$
{ Ime1=NovoIme();
Kod1=Generiraj(Kod2 || Ime1":=-"Ime2); }

$\langle E \rangle_{\text{Ime1}, \text{Kod1}} \rightarrow (\langle E \rangle_{\text{Ime2}, \text{Kod2}})$
{ Ime1=Ime2;
Kod1=Kod2; }

$\langle E \rangle_{\text{Ime1}, \text{Kod1}} \rightarrow \text{IDN}_{\text{Ime2}}$
{ Ime1=Ime2;
Kod1=Generiraj(""); }

Sintaksom upravljano generiranje međukôda

- Primjena atributne prijevodne gramatike
 - Generiranje troadresnih naredbi

$\langle S \rangle_{Kôd1} \rightarrow IDN_{Ime1} = \langle E \rangle_{Ime2, Kôd2}$
 $\{ Kôd1=Generiraj(Kôd2 || Ime1":=Ime2); \}$

$\langle E \rangle_{Ime1, Kôd1} \rightarrow \langle E \rangle_{Ime2, Kôd2} + \langle E \rangle_{Ime3, Kôd3}$
 $\{ Ime1=Novolme();$
 $Kôd1=Generiraj(Kôd2 || Kôd3 ||$
 $Ime1":=Ime2"+Ime3); \}$

$\langle E \rangle_{Ime1, Kôd1} \rightarrow \langle E \rangle_{Ime2, Kôd2} * \langle E \rangle_{Ime3, Kôd3}$
 $\{ Ime1=Novolme();$
 $Kôd1=Generiraj(Kôd2 || Kôd3 ||$
 $Ime1":=Ime2"*Ime3); \}$

⇒ $\langle S \rangle \xrightarrow{Kôd1, Ozn1, Ozn2} \text{while } \langle E \rangle_{lme1, Kôd2} \text{ do } \langle S \rangle \xrightarrow{Kôd3}$

```
{ Ozn1=NovaOznaka();
  Ozn2=NovaOznaka();
  Kôd1=Generiraj(
    Ozn1 ":" Kôd2
    "if" lme1"==0 goto" Ozn2
    Kod3
    "goto " Ozn1
  Ozn2 ":" ) }
```

