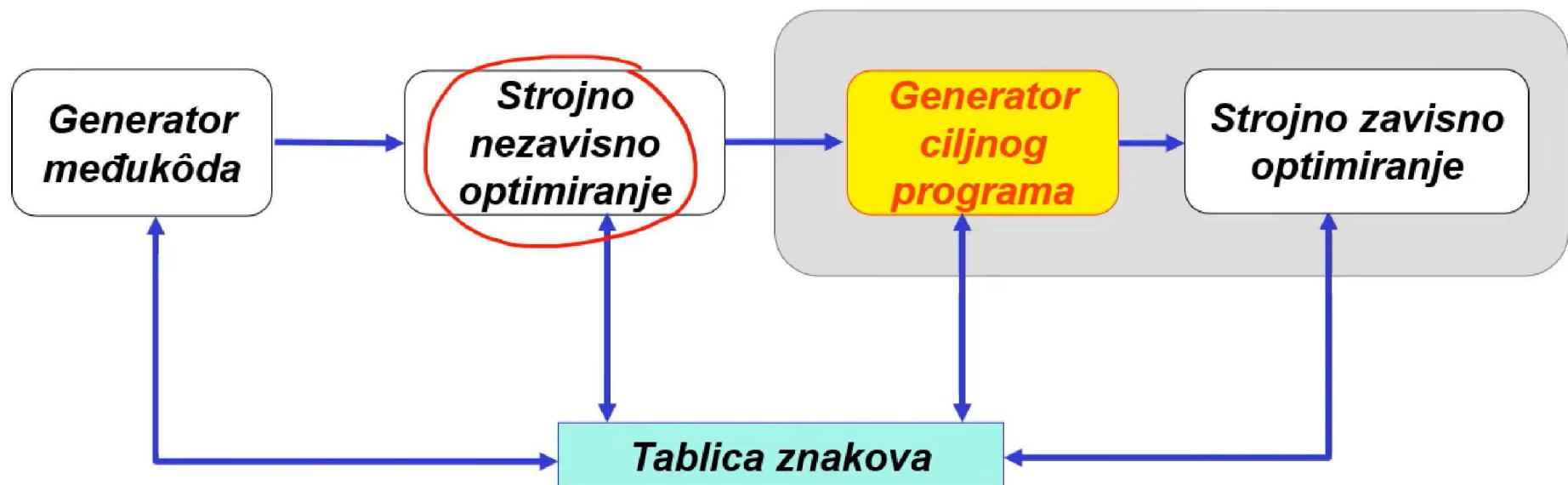


# Generiranje ciljnog programa



# Generiranje ciljnog programa

- Zahtjevi
  - ispravno i učinkovito generiranje ciljnog programa visoke kvalitete
- Mjera kvalitete
  - brzina njegovog izvođenja
  - veličina potrebnog memorijskog prostora
  - potrošena energija
- Visoka kvaliteta
  - učinkovita uporaba svih sredstava računala
    - registara, aritmetičko-logičkih jedinki, cjevovoda, priručne memorije

# Generiranje ciljnog programa

- **Zavisnost o**
  - arhitekturi računala, cilnjom jeziku, međukôdu i operacijskom sustavu
  - potrebno je
    - izabrati odgovarajući oblik i razinu međukôda
    - izabrati naredbe ciljnog jezika
    - izabrati redoslijed njihovog izvođenja
    - izraditi memorejske adrese
    - dodijeliti registre podacima
- **Složenost**
  - NP-teško ili NP-potpuno
  - heuristički postupci

# Struktura generatora ciljnog programa

• **Ulaz**

• **Izlaz**

• **Postupci**

• **Izrada adresa podacima**

- Izrada adresa podacima u statičkoj memoriji
- Izrada adresa podacima na stogu
- Izrada adresa podacima u gomili

• **Izrada adresa naredbama**

• **Izbor naredbe**

• **Upravljanje sadržajem registara**

• **Redoslijed izvođenja naredbi**

# Ulaz

- **Složenost postupaka generiranja**
  - ovisi o strukturi međukôda
  - **generiranje ciljnog programa na temelju**
    - troadresnih naredbi
    - postfiksнog i prefiksнog sustava oznaka
    - sažetog sintaksnog stabla
- **izravnavanje sintaksnog stabla**
  - u potpunosti je završeno za troadresne naredbe
  - djelomično za postfiksni i prefiksni sustav oznaka

- **Različiti ciljni jezici**
  - izvodljivi strojni jezici
  - premjestivi strojni jezici
  - mnemonički jezici
  - jezici virtualnih strojeva (**P-kôd i JAVA ByteCode**)
  - viši programski jezici

### ■ Izvodljivi ciljni program

- strojne naredbe procesora zapisane su nizom nula i jedinica
- memorijske adrese su u potpunosti izrađene
- spremi se u unaprijed određen memorijski prostor
- moguće ga je izvesti bez dodatne obrade
- izravno generiranje je složeno
  - zahtijeva istodobno prevođenje svih procedura u ciljni program
- koriste ga jednostavniji jezični procesori
  - interpretatori koji generiraju ciljni program izravno u radnu memoriju računala

- **Premjestivi ciljni program**
  - strojne naredbe zapisane su nizom nula i jedinica
  - adrese podataka i naredbi nisu u potpunosti izrađene
  - neizrađene adrese
    - relativni pomaci od nulte adrese
    - moguće ga je spremiti u bilo koji dio radne memorije
    - označe se sve naredbe koje nemaju u potpunosti izrađene adrese
    - vrijednosti adresa određuju se tijekom punjenja radne memorije
    - adresa se računa zbrajanjem relativnog pomaka i početne adrese dodijeljenog memorijskog prostora
  - **program punitelj**
    - izvodi postupak dorade adrese
  - omogućuje nezavisno prevođenje različitih procedura
  - **program povezivač**
    - povezuje zasebno prevedene procedure

- **Mnemonički ciljni program**
  - strojne naredbe zapisane mnemonicima
  - adrese naredbi i podataka su simboličke
  - makromnemonici
    - zamjenjuju unaprijed definirani niz od više strojnih naredbi
  - jednostavno generiranje
  - zahtijeva naknadno prevodenje u izvodljivi ciljni program primjenom mnemoničkog procesora

# Izrada adresa podacima

## IZVORNI PROGRAM

*Imena  
apstraktnih  
podataka*

## MEĐUKÔD

*Pokazivači  
na tablicu  
znakova*

## CILJNI PROGRAM

*Memorijske  
adrese  
podatkovnih  
objekata*

- Izrada memorijskih adresa ovisi o
  - strukturi podatkovnog objekta
  - organizaciji memorije
    - statička memorija, stog ili gomila
- Određivanje memorijskih adresa
  - na temelju vrijednosti parametara zapisanih u tablici znakova
  - proceduri se pridruže dva parametra
    - adresa početne naredbe procedure
    - adresa početka opisnika procedure

## Izrada adresa podacima u statickoj memoriji

- Nepromjenjive veličine
  - adresa početne naredbe procedure
  - adresa početka opisnika procedure
    - njihove vrijednosti određuju se na temelju
      - adrese početka staticke memorije
- izvodljivi ciljni program
  - vrijednosti parametara određuju se tijekom prevođenja izvornog programa u ciljni
- premjestivi ciljni program
  - vrijednosti parametara određuje program punitelj tijekom spremanja ciljnog programa u radnu memoriju
- mnemonički ciljni program
  - vrijednosti parametara određuju se tijekom obrade ciljnog programa mnemoničkim procesorom

## Izrada adresa podacima u statickoj memoriji

MOVE D0, (PočetakStatMem + PomakOpisnikaP17 + PomakStanja)

MOVE D1, (PočetakStatMem + PomakOpisnikaP17 + PomakStanja + 2)

CALL PočetakStatMem + PomakPotprogramaP17

MOVE (PočetakStatMem + PomakOpisnikaP17 + PomakStanja + 2), D1

MOVE (PočetakStatMem + PomakOpisnikaP17 + PomakStanja), D0

PočetakStatMem=1000, PomakOpisnikaP17=100,  
PomakStanja=10, PomakPotprogramaP17=500

|      |        |        |
|------|--------|--------|
| MOVE | D0     | (1110) |
| MOVE | D1     | (1112) |
| CALL | 1500   |        |
| MOVE | (1112) | D1     |
| MOVE | (1110) | D0     |

$$\underline{1000+100+10} = 1110$$

$$1000+100+10+2 = 1112$$

$$1000+500 = 1500$$

$$1000+100+10+2 = 1112$$

$$1000+100+10 = 1110$$

## Izrada adresa podacima na stogu

- **Adresa početka opisnika**
  - dinamički se mijenja tijekom izvođenja ciljnog programa
  - vrijednost adrese određuje se na temelju
    - vrijednosti kazaljke stoga
    - početnu vrijednost kazaljke stoga moguće je odrediti
      - tijekom analize izvornog programa
      - tijekom spremanja premjestivog ciljnog programa u radnu memoriju
      - tijekom obrade mnemoničkog ciljnog programa mnemoničkim procesorom
    - vrijednost adrese početka opisnika nanovo se računa tijekom svakog poziva procedure
  - **proširenje ciljnog programa**
    - naredbe koje računaju novu vrijednost adrese početka opisnika

## Izrada adresa podacima na stogu

- **A0**
  - adresni registar kazaljke stoga
- **A1**
  - tijekom poziva procedure
    - računa se nova vrijednost kazaljke stoga
  - završetak izvođenja procedure
    - dohvaća se stara vrijednost kazaljke stoga

## Izrada adresa podacima na stogu

MOVE A0, A1

Računa se nova vrijednost  
kazaljke stoga

ADD VeličinaOpisnikaP58, A1

MOVE A0, (A1 + PomakStanja)

MOVE A1, A0

MOVE D0, (A0 + PomakStanja + 2)

MOVE D1, (A0 + PomakStanja + 4)

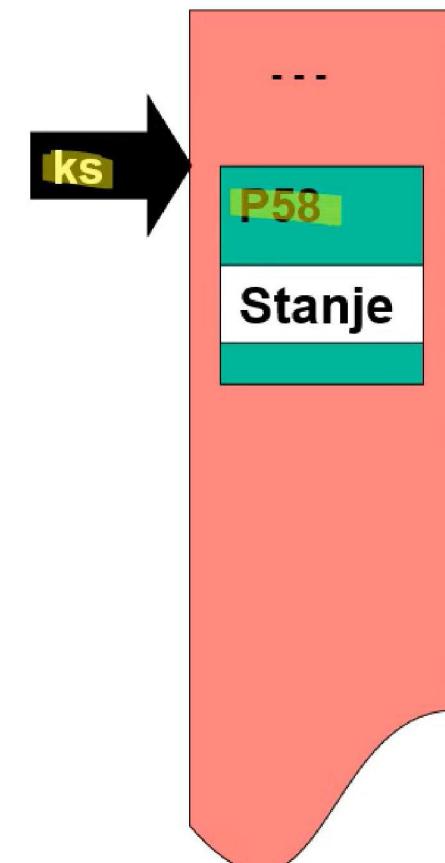
CALL PočetakStatMem + PomakPotprogramaP17

MOVE (A0 + PomakStanja + 4), D1

MOVE (A0 + PomakStanja + 2), D0

MOVE (A0 + PomakStanja), A1

MOVE A1, A0



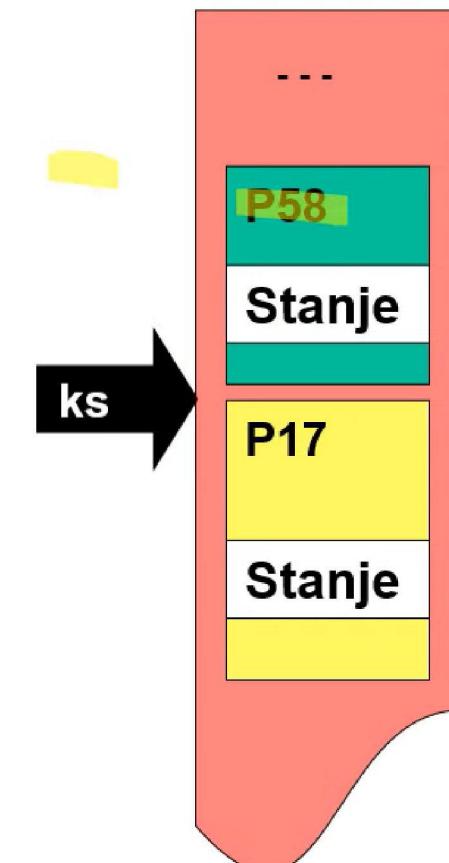
## Izrada adresa podacima na stogu

```
MOVE A0, A1  
ADD VeličinaOpisnikaP58, A1  
MOVE A0, (A1 + PomakStanja)  
MOVE A1, A0
```

Računa se nova vrijednost  
kazaljke stoga  
Sprema se stara vrijednost

```
MOVE D0, (A0 + PomakStanja + 2)  
MOVE D1, (A0 + PomakStanja + 4)  
CALL PočetakStatMem + PomakPotprogramaP17  
MOVE (A0 + PomakStanja + 4), D1  
MOVE (A0 + PomakStanja + 2), D0
```

```
MOVE (A0 + PomakStanja), A1  
MOVE A1, A0
```



## Izrada adresa podacima na stogu

**MOVE A0, A1**

**ADD VeličinaOpisnikaP58, A1**

**MOVE A0, (A1 + PomakStanja)**

**MOVE A1, A0**

Računa se nova vrijednost kazaljke stoga

Sprema se stara vrijednost  
Stavi u A0 novu vrijednost

**MOVE D0, (A0 + PomakStanja + 2)**

**MOVE D1, (A0 + PomakStanja + 4)**

**CALL PočetakStatMem + PomakPotprogramaP17**

**MOVE (A0 + PomakStanja + 4), D1**

**MOVE (A0 + PomakStanja + 2), D0**

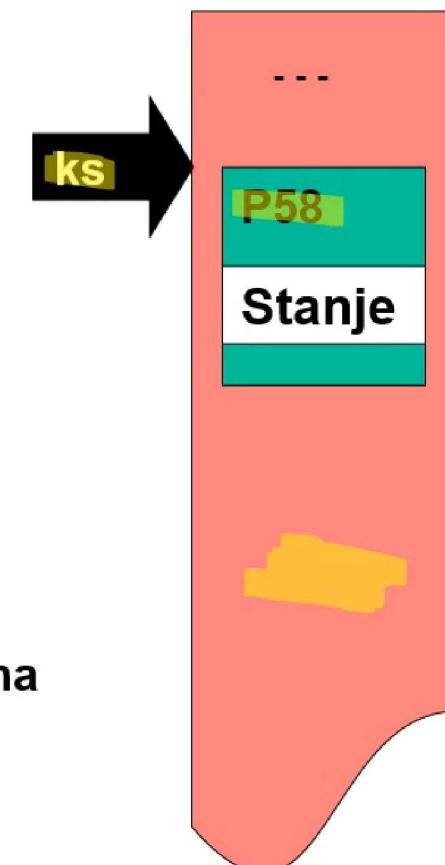
**MOVE (A0 + PomakStanja), A1**

**MOVE A1, A0**

Dohvaća se stara vrijednost kazaljke stoga

**Naredba LINK** - sprema staru vrijednost kazaljke stoga i računa novu tijekom poziva potprograma

**Naredba UNLINK** - obnavlja staru vrijednost kazaljke stoga tijekom završetka izvođenja potprograma



### ■ Opisnici potprograma

- spremaju se u segmente memorije koji se dinamički dodjeljuju
  - *DodjeliMem()*
  - *OslobodiMem()*
- **podatkovni registar D10**
  - razmjenjuje se veličina dinamički dodijeljenog segmenta
- **adresni registar A0**
  - vrijednost kazaljke koja pokazuje na dodijeljeni segment

# Izrada adresa podacima u gomili

```
MOVE      VeličinaOpisnikaP17, D10  
CALL      DodjeliMem  
MOVE      A0, (PočetakStatMem)
```

Dodjela memorije za opisnik

```
MOVE      (PočetakStatMem), A0  
MOVE      D0, (A0 + PomakStanja)  
MOVE      D1, (A0 + PomakStanja + 2)  
CALL      PočetakStatMem + PomakPotprogramaP17  
MOVE      (PočetakStatMem), A0  
MOVE      (A0 + PomakStanja + 2), D1  
MOVE      (A0 + PomakStanja), D0
```

```
MOVE      VeličinaOpisnikaP17, D10  
MOVE      (PočetakStatMem), A0  
CALL      OslobođiMem
```

Oslobađanje memorije

## Izrada adresa naredbama

- **Adrese naredbi**

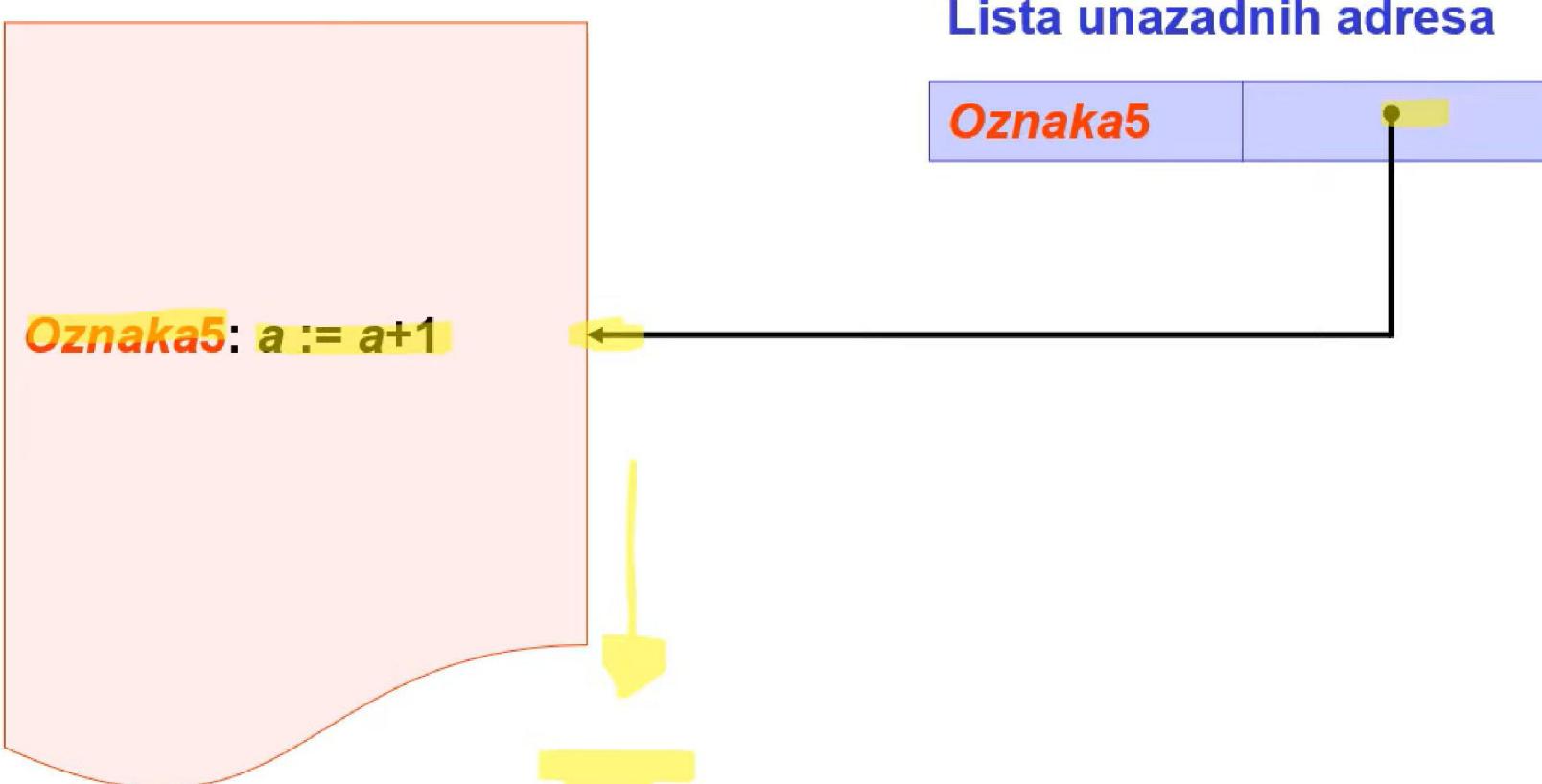
- **računaju se primjenom brojača**
- **veličina brojača povećava se za veličinu generirane naredbe izražene u oktetima**



- **Koriste se dvije liste:**

- **lista unazadnih adresa**
- **lista unaprijednih adresa**

# Izrada adresa naredbama



# Izrada adresa naredbama

skoči **Oznaka8**

Lista unaprijednih adresa

**Oznaka8**

**Oznaka Oznaka8 nije  
zadana u listi  
unazadnih adresa**

## Lista unazadnih adresa

Oznaka27

Oznaka27: skoči Oznaka31

## Lista unaprijednih adresa

- 1) U listu unazadnih adresa zapiše se oznaka **Oznaka27** i adresa naredbe grananja **Oznaka27: skoči Oznaka31**

svačl ozn 27

Oznaka27: skoči Oznaka31

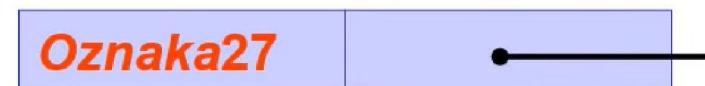
### Lista unazadnih adresa

Oznaka27



### Lista unaprijednih adresa

Oznaka27



- 2) U listi unaprijednih adresa traži se oznaka **Oznaka27**.

Ako se oznaka pronađe, onda se adresa naredbe grananja **Oznaka27: skoči Oznaka31** zapiše u sve naredbe na koje pokazuju kazaljke pridružene toj oznaci.

Zapis se briše iz liste unaprijednih adresa.

svočl Ozn 27

### Lista unazadnih adresa

Oznaka27

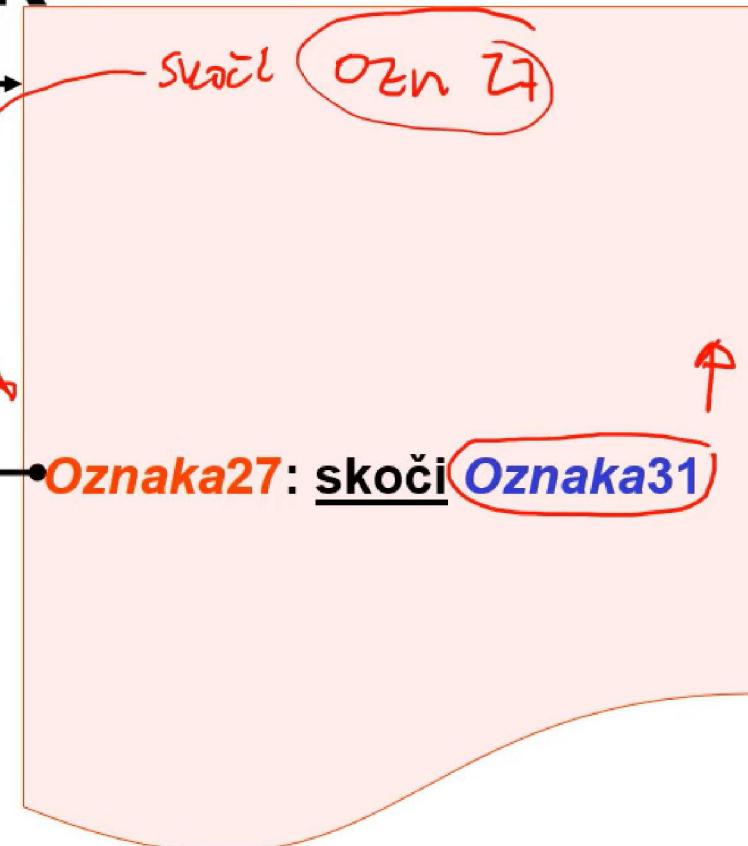
Oznaka27: skoči Oznaka31

### Lista unaprijednih adresa

- 2) U listi unaprijednih adresa traži se oznaka **Oznaka27**.

Ako se oznaka pronađe, onda se adresa naredbe grananja **Oznaka27: skoči Oznaka31** zapiše u sve naredbe na koje pokazuju kazaljke pridružene toj oznaci.

Zapis se briše iz liste unaprijednih adresa.



Lista unazadnih adresa

Oznaka31

Oznaka27

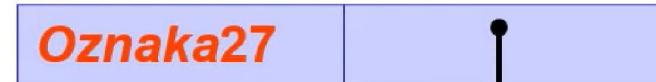
Lista unaprijednih adresa

- 3) U listi unazadnih adresa traži se oznaka **Oznaka31**.

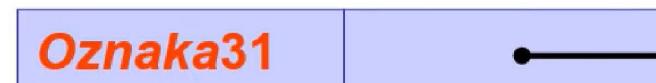
Ako se oznaka pronađe, onda je adresa odredišta naredbe grananja **Oznaka27: skoči Oznaka31** jednaka vrijednosti koja je pridružena toj oznaci u listi unazadnih adresa.



Lista unazadnih adresa



Lista unaprijednih adresa



- 4) Ne pronađe li se oznaka, pretražuje se lista unaprijednih adresa.

Ako se oznaka pronađe u listi unaprijednih adresa, onda se zapis pridružen toj oznaci proširi kazaljkom koja pokazuje na naredbu grananja **Oznaka27:skoči Oznaka31**.

Ne pronađe li se oznaka, lista se proširi novim zapisom: oznakom **Oznaka31** i kazaljkom koja pokazuje na naredbu grananja **Oznaka27:skoči Oznaka31**.

## Izbor naredbe

- **Jednolik skup naredbi**
  - naredbe ciljnog programa podjednake su veličine i trajanja
    - jednostavno generiranje ciljnog programa
      - izbor naredbe ne ovisi o njezinoj veličini i brzini
- **Potpun skup naredbi ciljnog programa**
  - naredbama omogućen pristup svim sredstvima procesora
    - svim registrima, načinima adresiranja, podatkovnim objektima, itd.
  - jednostavno generiranje ciljnog programa
    - nije potrebno definirati zasebna pravila generiranja ciljnog programa

## Izbor naredbe

Var1 := Var2+1

- **Jednolik skup naredbi**
  - **naredba zbrajanja ADD i inkrementiranja INC**  
—jednake veličine i trajanja
- **Potpun skup naredbi**
  - omogućena je uporaba svih načina adresiranja

**MOVE**

*MemAdrPribrojnik1, MemAdrZbroj*

**ADD**

*MemAdrPribrojnik2, MemAdrZbroj*

## Izbor naredbe

**Var1 := Var2+1**

- Ako se u nastavku programa koristi jedan od pribrojnika ili zbroj
  - generiraju se strojne naredbe koje dohvaćaju operandе u registre

|      |  |
|------|--|
| MOVE | <i>MemAdrPribrojnik1, RegistarPribrojnika1</i> |
| MOVE | <i>MemAdrPribrojnik2, RegistarPribrojnika2</i> |
| MOVE | <i>RegistarPribrojnika1, RegistarZbroja</i>    |
| ADD  | <i>RegistarPribrojnika2, RegistarZbroja</i>    |
| MOVE | <i>RegistarZbroja, MemAdrZbroj</i>             |

## Izbor naredbe

Var1 := Var2+1

### ■ Naredba INC

- manje je veličine od naredbe ADD
- izvodi se u manjem broju procesorskih ciklusa
- koristi samo jedan operand
  - naredba zbrajanja ADD koristi dva operanda
- Određivanje vrijednosti pribrojnika
  - jednostavno za konstantu, ali ne za varijablu
- Ako je omogućena uporaba vrijednosti jednog od pribrojnika
  - umjesto naredbe zbrajanja ADD generira se naredba inkrementiranja INC

# Upravljanje sadržajem registara

Memorijska  
hijerarhija

Procesor



Registri procesora

Više razina  
priručnih memorija

Radna memorija  
računala

Jedinke masovne  
memorije  
(na primjer,  
magnetski disk)

# Upravljanje sadržajem registara

## RISC procesori

- međusobno jednaki registri koji se ne razlikuju po načinu uporabe

### dvije vrste registara

- namijenjeni cijelobrojnim podacima
- namijenjeni podacima s posmačnim zarezom



- upravljanje sadržajem registara RISC procesora

- nije zahtjevni problem kao upravljanje sadržajem registara CISC procesora

## CISC procesori

- potrebno je znati način uporabe svakog od registara posebne namjene

- samo neke registre moguće je koristiti kao kazaljke stoga

- većini naredbi CISC procesora nije omogućen pristup svim registrima, već samo odabranima

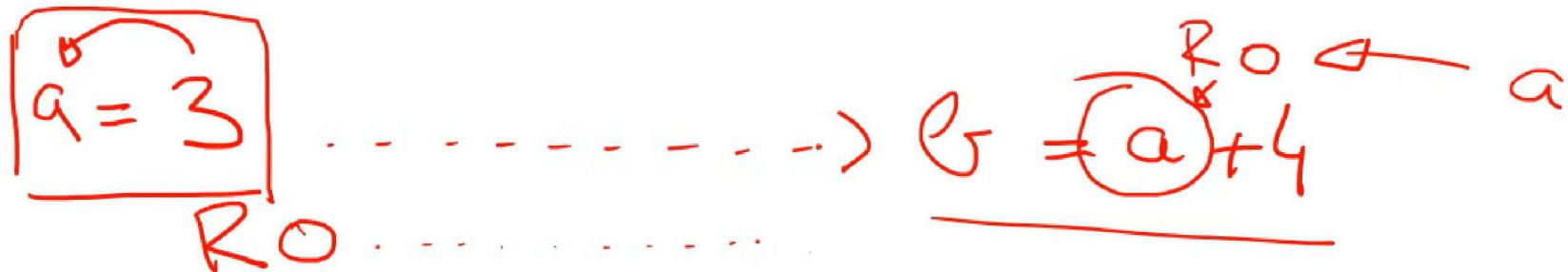
# Upravljanje sadržajem registara

- **Operacijski sustav**
  - zahtijeva isključivu uporabu pojedinih registara ili im namjenjuje zasebnu ulogu
  - registre je potrebno koristiti u skladu s ulogom koja im je namijenjena

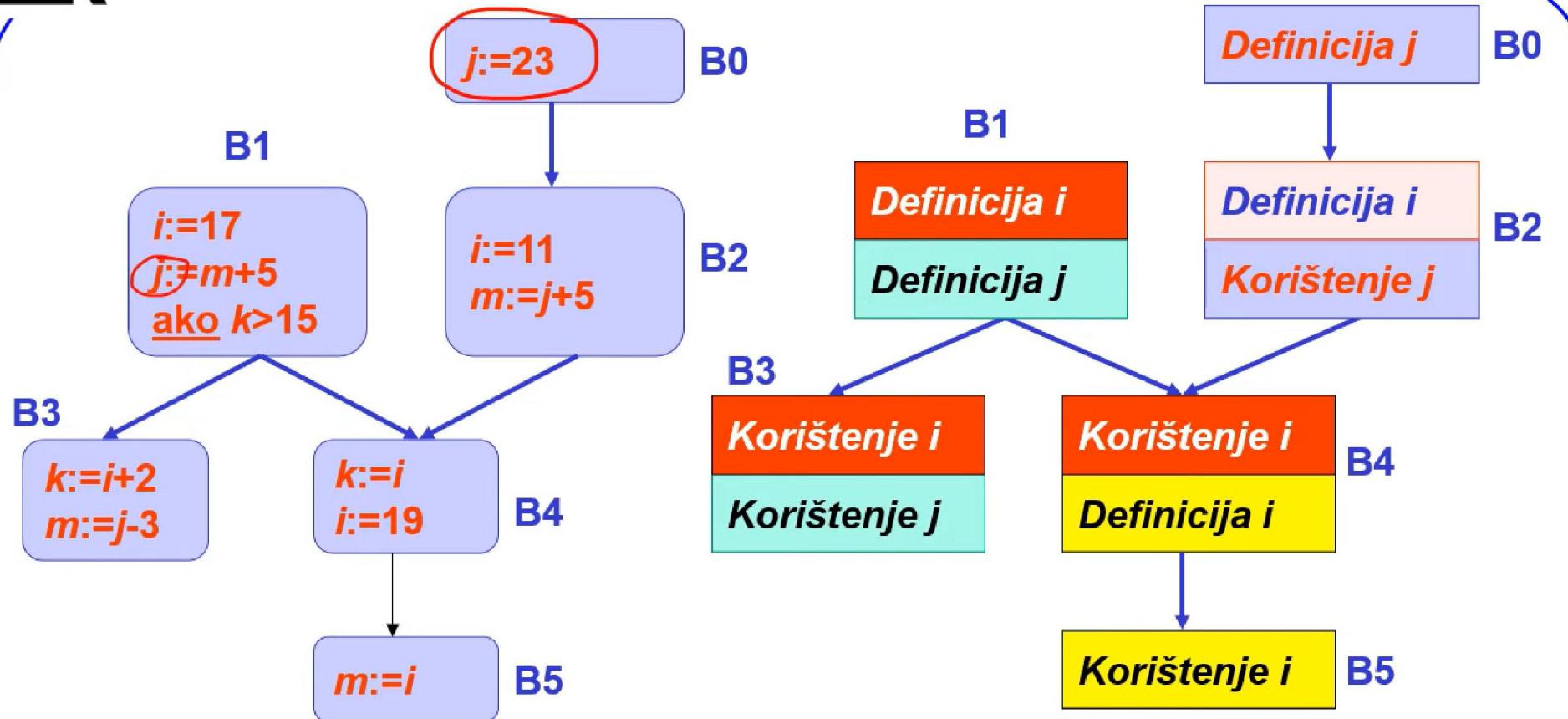
# Upravljanje sadržajem registara

- **Potrebno je**
  - **odrediti koji podaci se spremaju u registre**
  - **pojedinim podacima dinamički dodijeliti register**
- **Podaci koji se spremaju registre**
  - **varijable, privremene varijable, konstante**
  - **vrijednosti stavki složenih apstraktnih podataka ne stavljaju se u registre**
  - **one se dohvaćaju izravno iz memorije**
  - **dodatni postupci optimiranja**
    - označe određene stavke polja imenima skalarnih varijabli
    - vrijednost skalarne varijable moguće je spremiti u register
    - omogućuje pristup istom podatku na više različitih načina
      - na primjer primjenom indeksa polja i primjenom imena skalarne varijable

# Upravljanje sadržajem registara



- **Definicija/uporaba lanac (du-lanac)**
  - čine naredbe programa koje povezuje
    - mjesto definicije vrijednosti varijable
    - sva mesta njezine uporabe
  - za svaku definiciju varijable definira se zasebni du-lanac

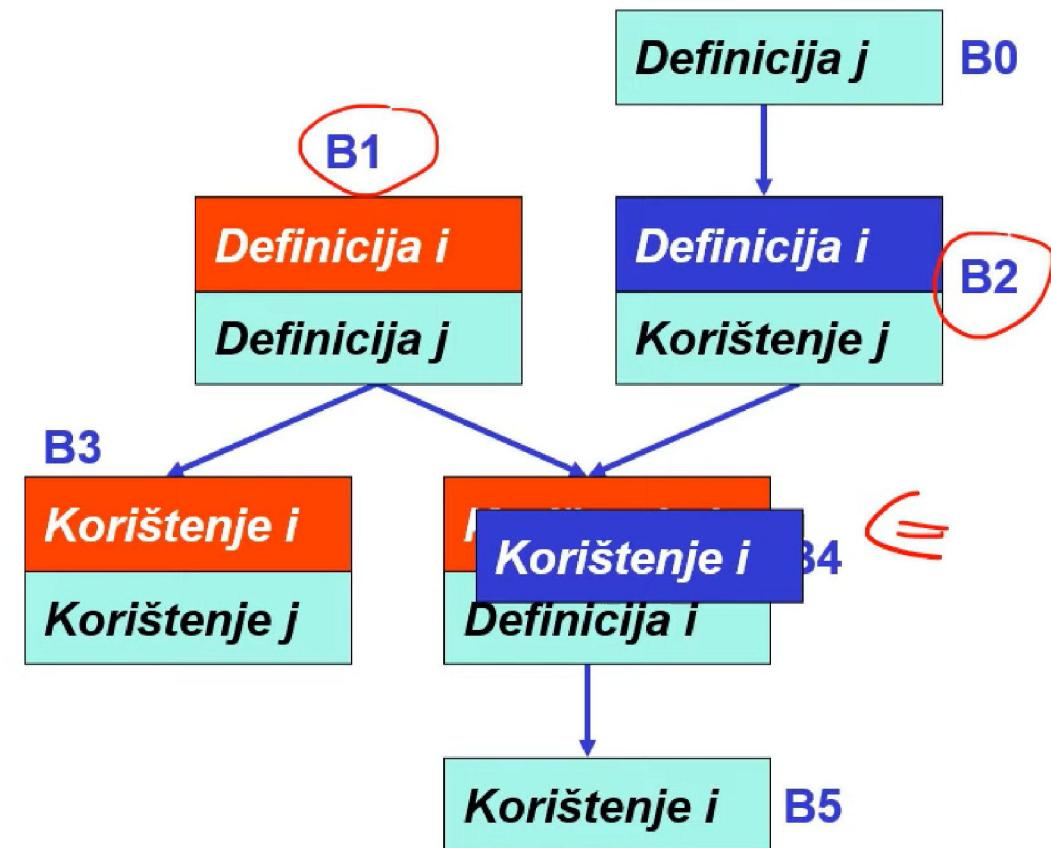


- du<sub>5</sub>** Definicija varijable  $j$  u bloku B0, uporaba varijable  $j$  u bloku B2
- du<sub>2</sub>** Definicija varijable  $i$  u bloku B2, uporaba varijable  $i$  u bloku B4
- du<sub>1</sub>** Definicija varijable  $i$  u bloku B1, uporaba varijable  $i$  u blokovima B3 i B4
- du<sub>4</sub>** Definicija varijable  $j$  u bloku B1, uporaba varijable  $j$  u bloku B3
- du<sub>3</sub>** Definicija varijable  $i$  u bloku B4, uporaba varijable  $i$  u bloku B5

# Upravljanje sadržajem registara

- Mrežica**

- unija *du-lanaca iste varijable* koji imaju zajedničkih naredbi



- M<sub>1</sub>** Definicija varijable *i* u blokovima B1 i B2, uporaba varijable *i* u blokovima B3 i B4
- M<sub>2</sub>** Definicija varijable *i* u bloku B4, uporaba varijable *i* u bloku B5
- M<sub>3</sub>** Definicija varijable *j* u bloku B1, uporaba varijable *j* u bloku B3
- M<sub>4</sub>** Definicija varijable *j* u bloku B0, uporaba varijable *j* u bloku B2

# Upravljanje sadržajem registara

$$M_A \Leftrightarrow M_B$$

- **Graf zavisnosti mrežica**

- čvorovi – mrežice
- grane - međusobna zavisnost mrežica
  - zajedničke naredbe
  - isključuju se naredbe koje istodobno definiraju vrijednost varijable jedne mrežice, a koriste vrijednost definiranu u drugoj mrežici

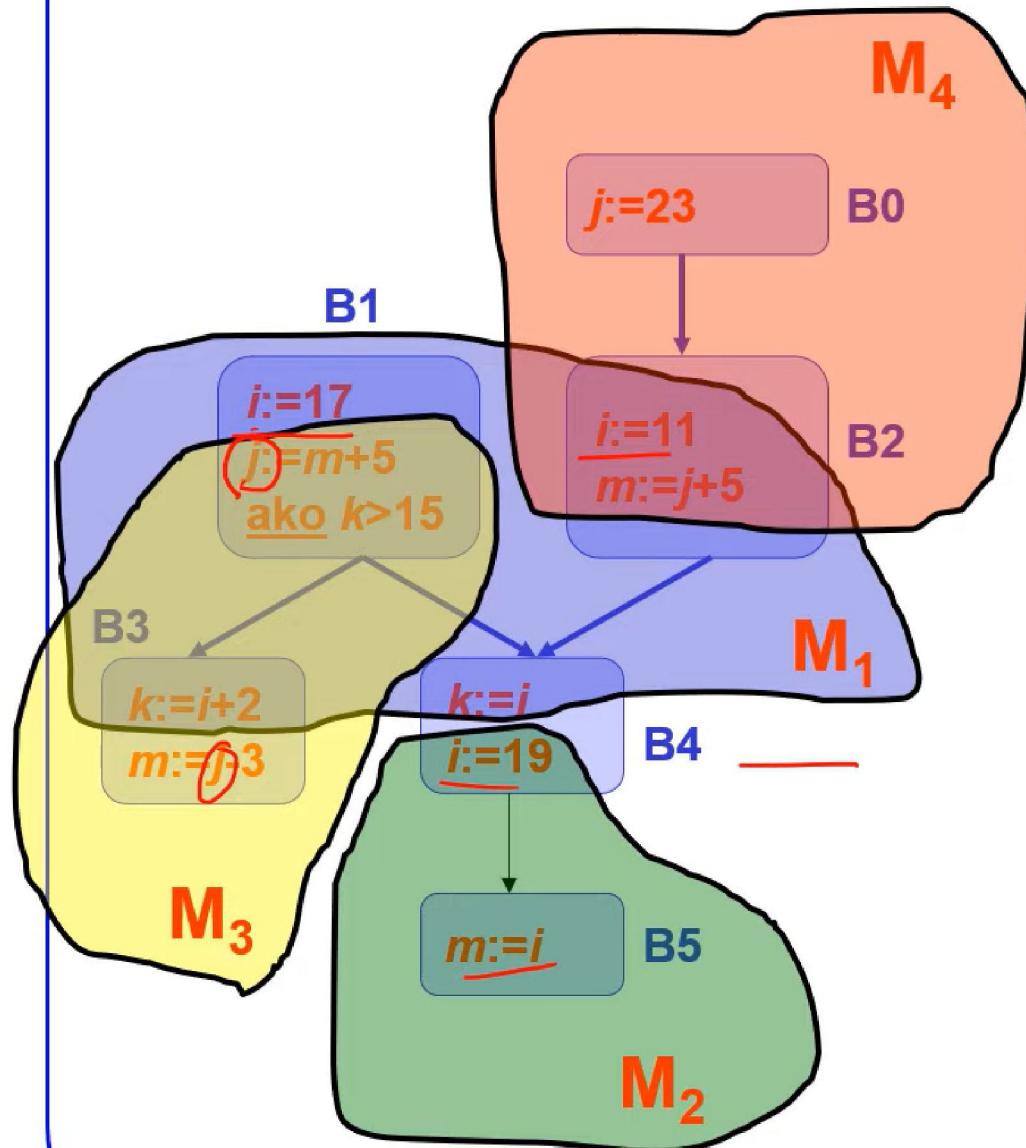


$$\overline{B} = \overline{A} + 3$$

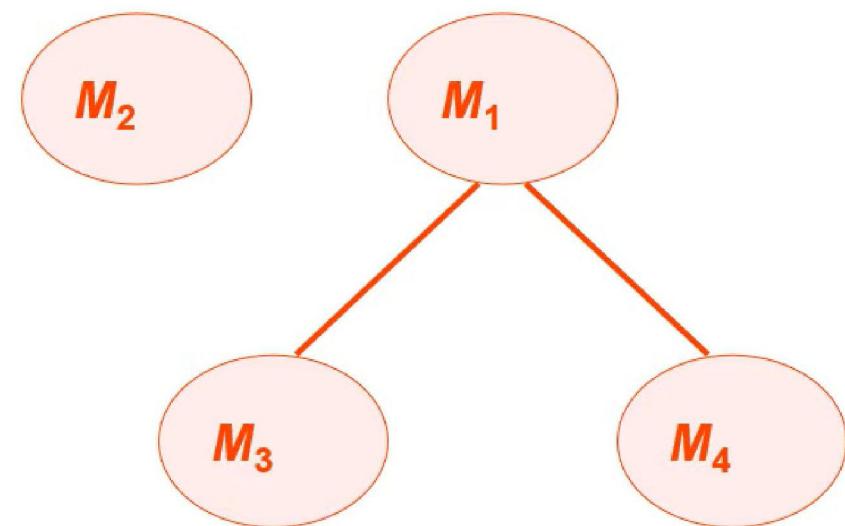
- **Susjedi**

- čvorovi povezani granom
- $m$  susjeda -  $m$ -tog stupnja

# Upravljanje sadržajem registara



## Graf zavisnosti mrežica



# Upravljanje sadržajem registara

- **Bojanje čvorova grafa zavisnosti**
  - algoritam je prvi opisao John Cocke 1971. godine
  - prva praktična primjena ostvarena je 1981. godine u jezičnom procesoru programskog jezika PL/I računala IBM 370
  - kasnije u jezičnom procesoru PL.8 RISC računala IBM 801
  - algoritam se izvodi u pet koraka

# Upravljanje sadržajem registara

## 1) Odrede se mrežice

- (1)  $i := 11$
- (2)  $j := 5$
- (3)  $m := m - m$
- (4)  $k := i * 10$
- (5)  $n := i - j$
- (6)  $i := k + 12$

- $M_1$  Definicija  $i$  u naredbi (1), uporaba u naredbama (4) i (5)
- $M_2$  Definicija  $j$  u naredbi (2), uporaba u naredbi (5)
- $M_3$  Definicija  $m$  u naredbi (3)
- $M_4$  Definicija  $k$  u naredbi (4), uporaba u naredbi (6)
- $M_5$  Definicija  $n$  u naredbi (5)
- $M_6$  Definicija  $i$  u naredbi (6)

# Upravljanje sadržajem registara

## 2) Mrežicama se dodijele simbolički registri

- (1)  $i := 11$
- (2)  $j := 5$
- (3)  $m := m - m$
- (4)  $k := i * 10$
- (5)  $n := i - j$
- (6)  $i := k + 12$

**M<sub>1</sub>** - **r<sub>1</sub>** - **i**

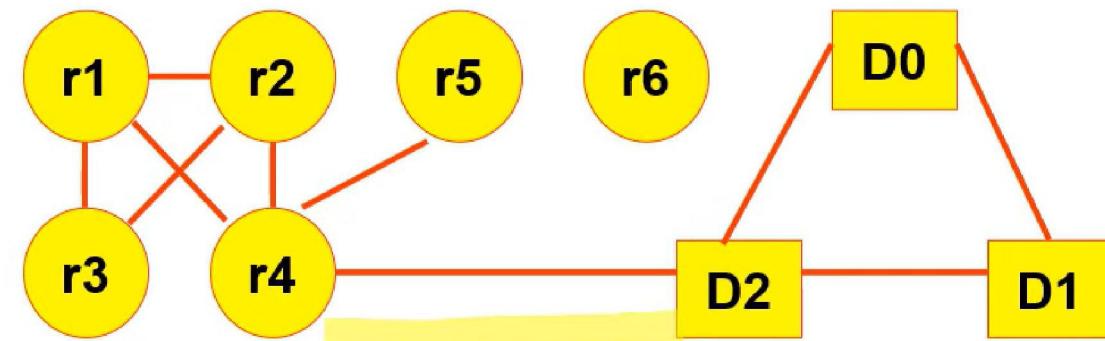
## 2) Mrežicama se dodijele simbolički registri

- (1)  $r1 := 11$
- (2)  $r2 := 5$
- (3)  $r3 := r3 - r3$
- (4)  $r4 := r1 * 10$
- (5)  $r5 := r1 - r2$
- (6)  $r6 := r4 + 12$

|       |   |      |   |     |
|-------|---|------|---|-----|
| $M_1$ | - | $r1$ | - | $i$ |
| $M_2$ | - | $r2$ | - | $j$ |
| $M_3$ | - | $r3$ | - | $m$ |
| $M_4$ | - | $r4$ | - | $k$ |
| $M_5$ | - | $r5$ | - | $n$ |
| $M_6$ | - | $r6$ | - | $i$ |

### 3) Gradi se graf zavisnosti simboličkih i stvarnih registara

- |     |                 |
|-----|-----------------|
| (1) | $r1 := 11$      |
| (2) | $r2 := 5$       |
| (3) | $r3 := r3 - r3$ |
| (4) | $r4 := r1 * 10$ |
| (5) | $r5 := r1 - r2$ |
| (6) | $r6 := r4 + 12$ |



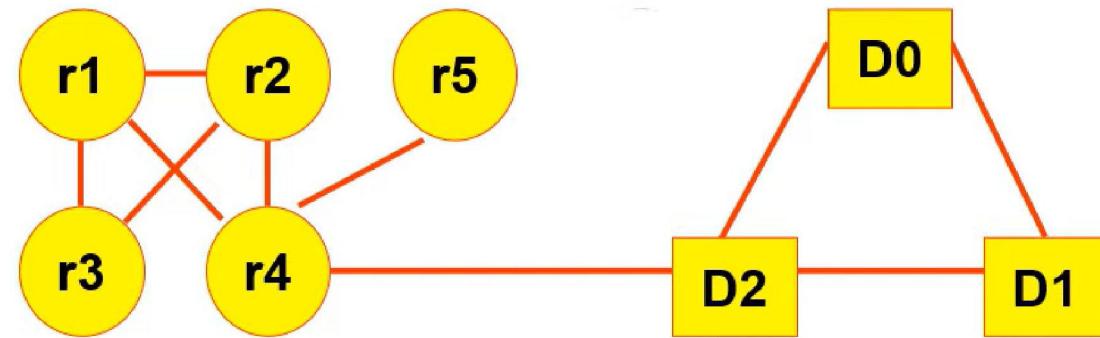
## Upravljanje sadržajem registara

3) Ako generator raspolaže s **R** stvarnih registara procesora, onda se čvorovi grafa zavisnosti boje s **R** različitih boja tako da dva susjedna čvora nisu obojana istom bojom.

- **NP-potpuni problem**
- koriste se heuristički postupci
- Chaitin - najviše korišteni heuristički postupak

# Upravljanje sadržajem registara

$|x| < 3$



(r5, {r4})

(r6, { })

# Upravljanje sadržajem registara

$|x| < \textcircled{3}$

|                |
|----------------|
| (r1, {r2, r4}) |
| (r3, {r1, r2}) |
| (r5, {r4})     |
| (r6, { })      |

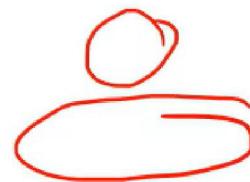


# Upravljanje sadržajem registara

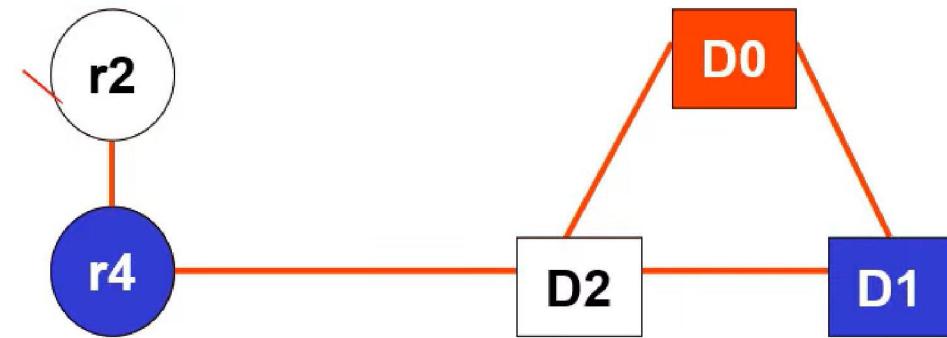
|                |
|----------------|
| (D1, { })      |
| (D0, {D1})     |
| (D2, {D0, D1}) |
| (r4, {D2})     |
| (r2, {r4})     |
| (r1, {r2, r4}) |
| (r3, {r1, r2}) |
| (r5, {r4})     |
| (r6, { })      |

D1

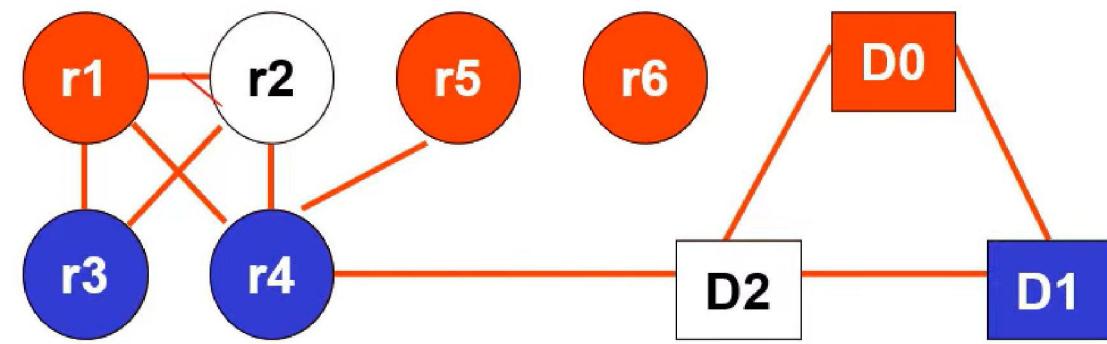
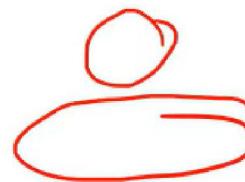
# Upravljanje sadržajem registara



|                |
|----------------|
| (r1, {r2, r4}) |
| (r3, {r1, r2}) |
| (r5, {r4})     |
| (r6, { })      |



# Upravljanje sadržajem registara

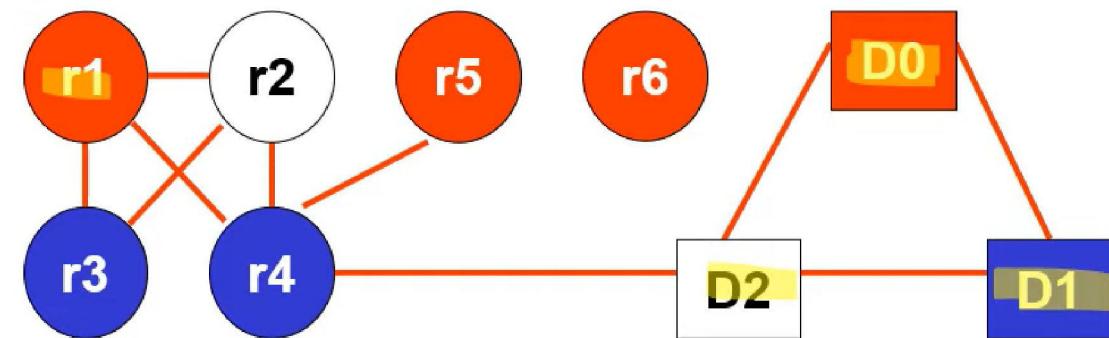


**(r6, { })**

# Upravljanje sadržajem registara

5) Stvarni register dodijeli se onim simboličkim registrima koji su obojani istom bojom.

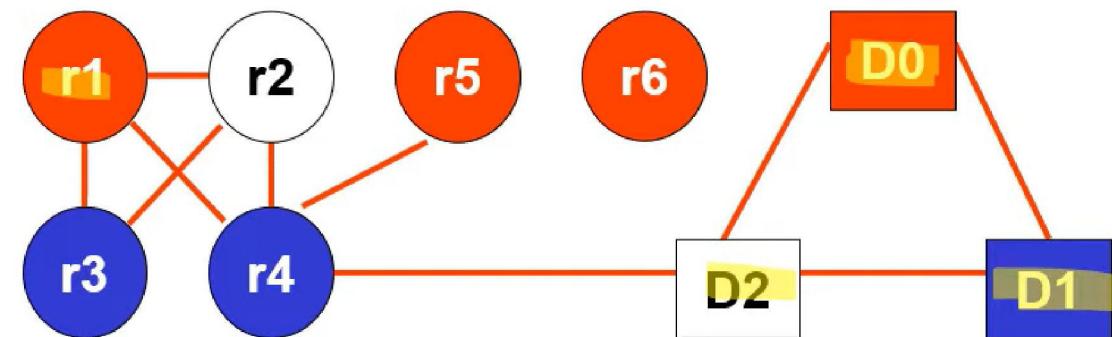
|     |                 |
|-----|-----------------|
| (1) | $r1 := 11$      |
| (2) | $r2 := 5$       |
| (3) | $r3 := r3 - r3$ |
| (4) | $r4 := r1 * 10$ |
| (5) | $r5 := r1 - r2$ |
| (6) | $r6 := r4 + 12$ |



# Upravljanje sadržajem registara

5) Stvarni register dodijeli se onim simboličkim registrima koji su obojani istom bojom.

|     |                 |
|-----|-----------------|
| (1) | $D0 := 11$      |
| (2) | $D2 := 5$       |
| (3) | $D1 := D1 - D1$ |
| (4) | $D1 := D0 * 10$ |
| (5) | $D0 := D0 - D2$ |
| (6) | $D0 := D1 + 12$ |



## Redoslijed izvođenja naredbi

- **Izborom odgovarajućeg redoslijeda izvođenja naredbi**
  - smanjuje se veličina ciljnog programa
  - skraćuje se vrijeme izvođenja
  - učinkovito se koriste
    - registri procesora
    - priručna memorija
    - **naredbeni cjevovod**
    - naredbeni registri
    - aritmetičko-logičke jedinke
- **naredbeni cjevovod**
  - u svakom ciklusu moguće je pokrenuti izvođenje jedne nezavisne naredbe
  - **prazni ciklus - zaustavlja se slijedno izvođenje naredbi**
    - zavisne naredbe
    - naredbe koje u istom ciklusu koriste ista sredstva procesora

## Redoslijed izvođenja naredbi

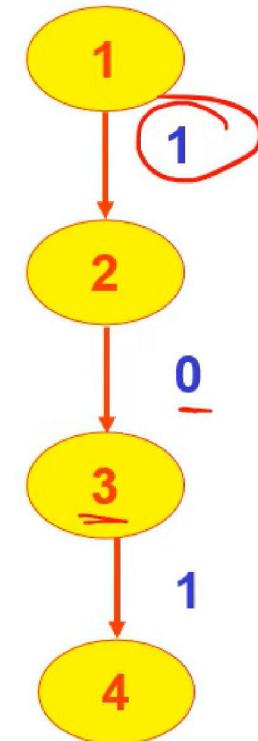
- **Učinkovita uporaba različitih sredstava računala**
  - nije moguće uvijek postići
  - **učinkovita uporaba registara**
    - grupiraju se naredbe koje koriste iste podatke - zavisne
  - **učinkovita uporaba naredbenog cjevovoda**
    - grupiraju se nezavisne naredbe
  - **suprotni zahtjevi**

Međukôd

- $\Rightarrow a := a + 1$
- $b := b + 1$
- (1) MOVE a, D0
  - (2) INC D0
  - (3) MOVE D0, a
  - (4) MOVE b, D0
  - (5) INC D0
  - (6) MOVE D0, b

Ciljni  
program

DO



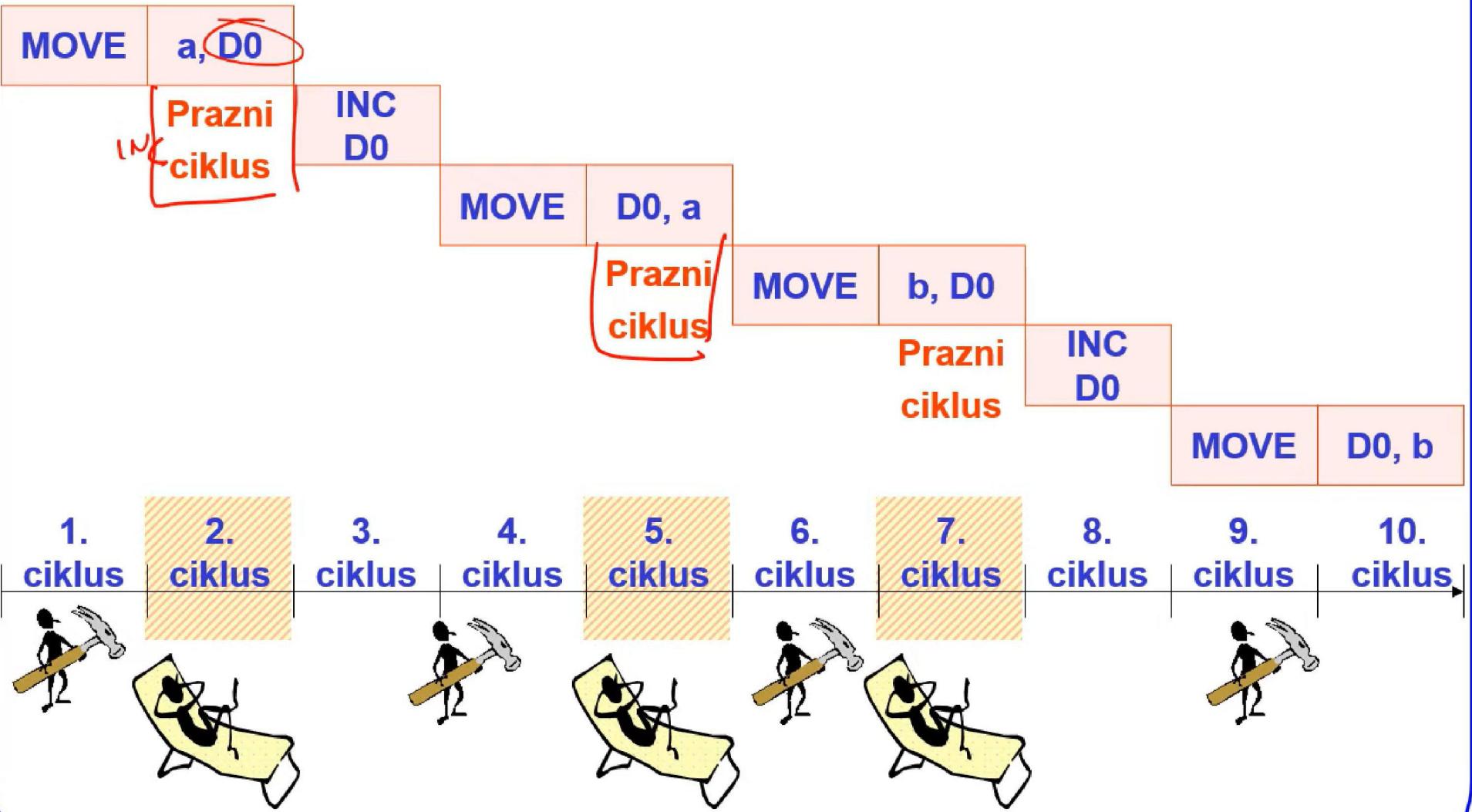
Naredba  
iz memorije

Dohvat naredbe ✓  
Dekodiranje naredbe ✓  
Izvođenje registrarskih naredbi ✓

Pristup  
podacima iz  
memorije ✓

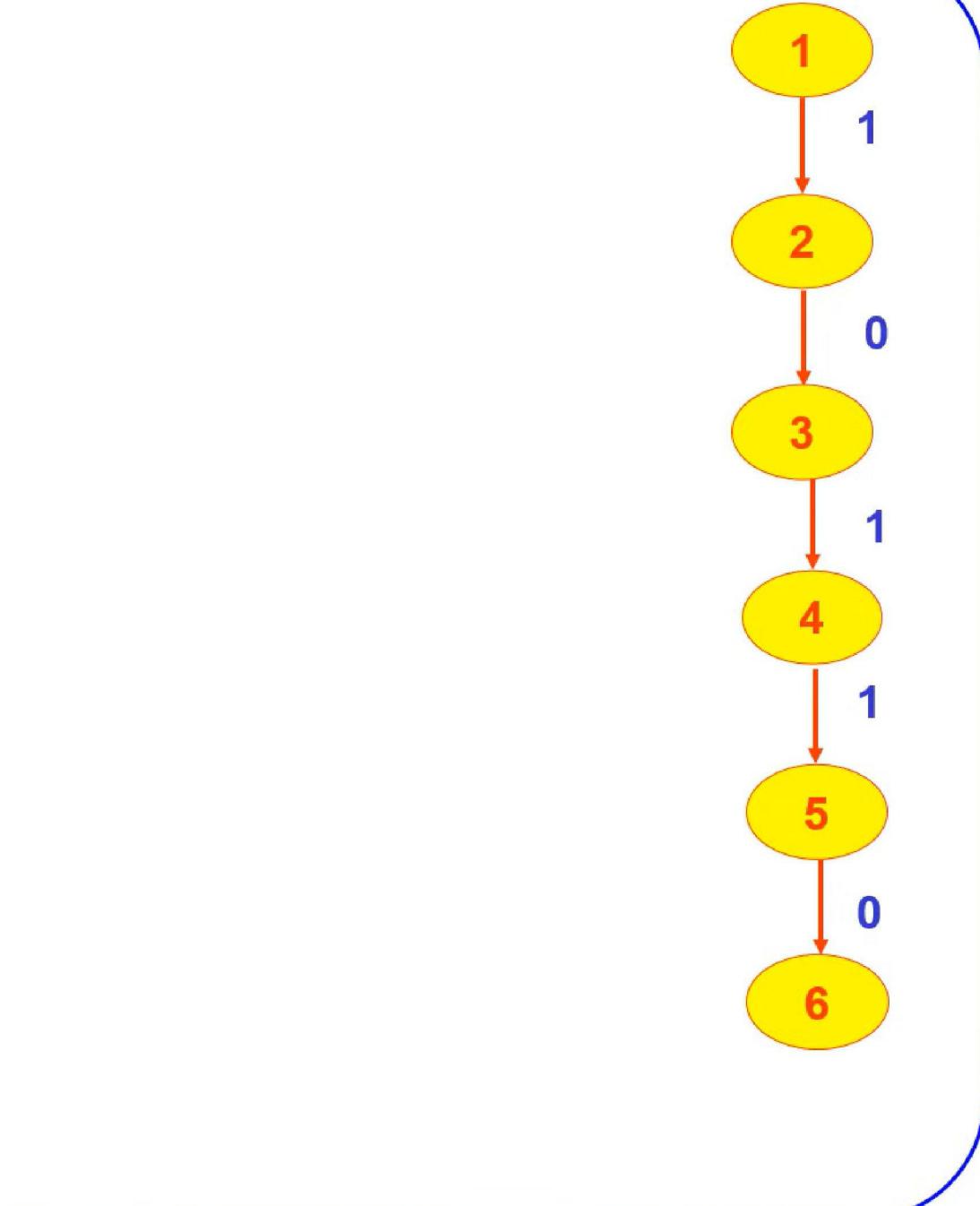
1. ciklus

2. ciklus



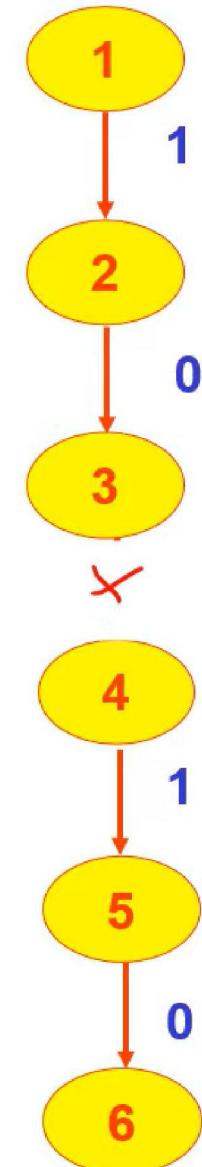
**Ciljni  
program**

- (1) MOVE a, D0
- (2) INC D0
- (3) MOVE D0, a
- (4) MOVE b, D1
- (5) INC D1
- (6) MOVE D1, b



**Ciljni  
program**

- (1) MOVE a, D0
- (2) INC D0
- (3) MOVE D0, a
- (4) MOVE b, D1
- (5) INC D1
- (6) MOVE D1, b



**Ciljni  
program**

- (1) MOVE a, D0
- (2) INC D0
- (3) MOVE D0, a
- (4) MOVE b, D1
- (5) INC D1
- (6) MOVE D1, b

**Ciljni  
program**

- (1) MOVE a, D0
- (4) MOVE b, D1
- (2) INC D0
- (5) INC D1
- (3) MOVE D0, a
- (6) MOVE D1, b

