

# Ostvarenje apstraktnih tipova podataka izvornog jezika podatkovnim objektima ciljnog jezika

- Apstraktni tipovi podataka izvornog jezika
- Podatkovni objekti ciljnog jezika
- Načini ostvarivanja apstraktnih tipova podataka



# Apstraktni tipovi podataka izvornog jezika

- **Apstraktni podaci**

- Podatkovne stavke
  - podatkovne stavke su apstraktni podaci
  - hijerarhijska definicija apstraktnih podataka
- Skup pravila
- Skup ograničenja nad relacijama koje povezuju podatkovne stavke
- Primjeri apstraktnih tipova podataka
  - niz, red, stog, stablo, usmjereni graf, polje, tablica, itd.

# Apstraktni tipovi podataka izvornog jezika

- **Niz**
  - podatkovne stavke su slijedno uređene
  - slijedno pretraživanje niza
  - nizovi se nadovezuju, uspoređuju, dijele, itd.
- **Red i stog**
  - dinamički se mijenjaju dohvatom sadržaja stavke i dodavanjem nove stavke
  - **Red**
    - stavke se dodaju na kraj reda
    - stavke se dohvaćaju s početka reda
    - First-In-First-Out* struktura
  - **Stog**
    - dohvaća se ona stavka koja je posljednja dodana na stog
    - Last-In-First-Out* struktura



# Apstraktni tipovi podataka izvornog jezika

- **Stablo**
  - stavkama se dodaju *kazaljke*
  - kazaljke pokazuju na stavke niže hijerarhijske razine
  - stavka najviše hijerarhijske razine je *korijen stabla*
  - listovi stabla su stavke na najnižoj hijerarhijskoj razini
  - kazaljke listova pokazuju na strukture podataka koje nisu dio stabla
  - **primjena stabla**
    - definicija procesa parsiranja, opis aktiviranja procedura, itd.
- **Usmjereni graf**
  - moguće je da kazaljka stavke niže hijerarhijske razine
    - pokazuje na stavku više hijerarhijske razine
  - **primjena usmjerenog grafa**
    - za prikaz složenih tipova obilježja, međukôda, itd.

# Apstraktni tipovi podataka izvornog jezika

- **Polje**
  - Skup stavki jednoznačno uređenih skupom cijelih brojeva
  - Skup cijelih brojeva
    - određuje mjesto stavke u polju
    - omogućuje njezin izravni dohvati
  - $A[i_1, i_2, \dots, i_n]$ 
    - $[i_1, i_2, \dots, i_n]$  uređeni skup cijelih brojeva
    - indeksi polja
  - $n$  indeksa
    - $n$ -dimenzionalno polje
  - **Primjer:**
    - vrijednosti prvog indeksa
    - vrijednosti drugog indeksa
    - vrijednosti trećeg indeksa
    - $A[1:20, 0:7, 3:17]$
    - cijeli brojevi od 1 do 20
    - cijeli brojevi od 0 do 7
    - cijeli brojevi od 3 do 17
  - **Veličina područja indeksa**
    - kardinalni broj skupa njegovih cjelobrojnih vrijednosti
    - veličine područja indeksa polja  $A$  su 20, 8 i 15



# Apstraktni tipovi podataka izvornog jezika

- **Tablice**

- **Stavkama tablice dodjeljuju se ključevi**
- **Ključevi omogućuju**
  - dohvat stavke
  - dodavanje stavke u tablicu
- **Pristupnom se mehanizmu predaju podaci**
  - na temelju podatka odredi se ključ
  - na temelju ključa izračuna se kazaljka koja pokazuje na mjesto zapisa stavke
- **Različiti načini određivanja ključa i kazaljke koja pokazuje na stavku**
  - binarno pretraživanje, raspršeno adresiranje, itd.



## Podatkovni objekti ciljnog jezika

- **Podatkovni objekti ciljnog jezika**
  - **Spremaju se izravno u memoriju**
    - memorija je niz slijednih ćelija kojima se pristupa primjenom njihove adrese
  - **Izravno ovise o slijednoj strukturi memorije**
  - **Element**
    - najjednostavniji podatkovni objekt
    - niz slijednih i susjednih memorijskih ćelija
    - dijelovi elementa
      - polja u koja se spremaju korisnički podaci
      - polja koja se koriste za gradnju složenih podatkovnih objekata
    - uloga elementa
      - slična ulozi stavke
      - element nema složenu strukturu

# Podatkovni objekti ciljnog jezika

- **Vektor**

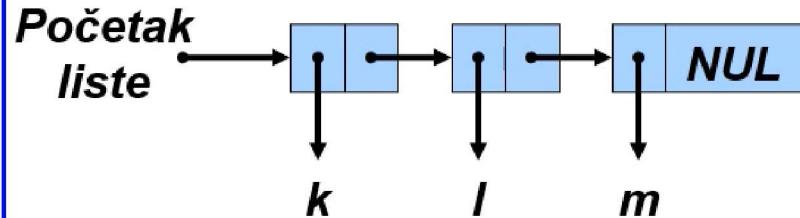
- Skup elemenata spremljenih u fizički susjedne memorijske ćelije
- Elementi jednake veličine
  - vektor je definiran s tri parametra
    - adresa početnog elementa
    - veličina jednog elementa
    - broj elemenata
- Sličan apstraktnom podatku jednodimenzionalnog polja
  - označava se istom oznakom
    - $V[1:n]$

- **Liste**

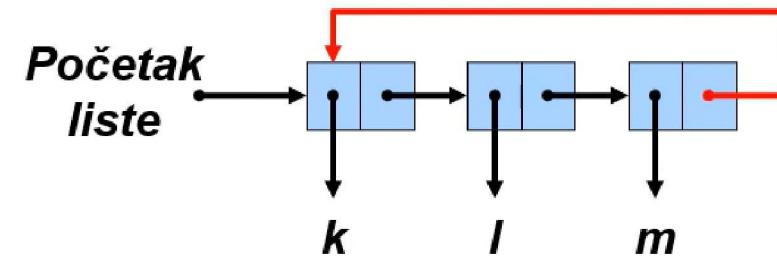
- Skup elemenata povezanih kazaljkama
- Jednostruko i dvostruko povezane
- Linearno i kružno povezane



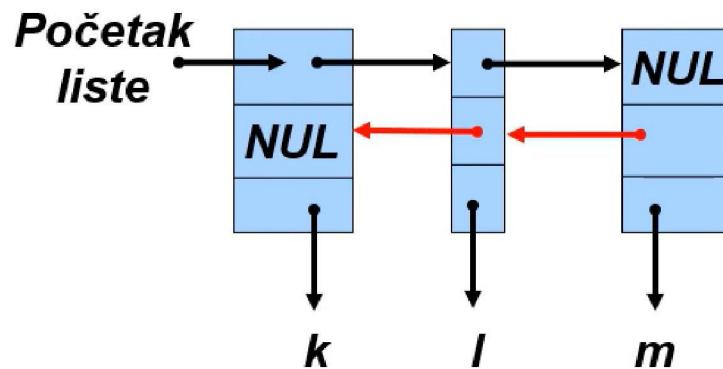
# Podatkovni objekti ciljnog jezika



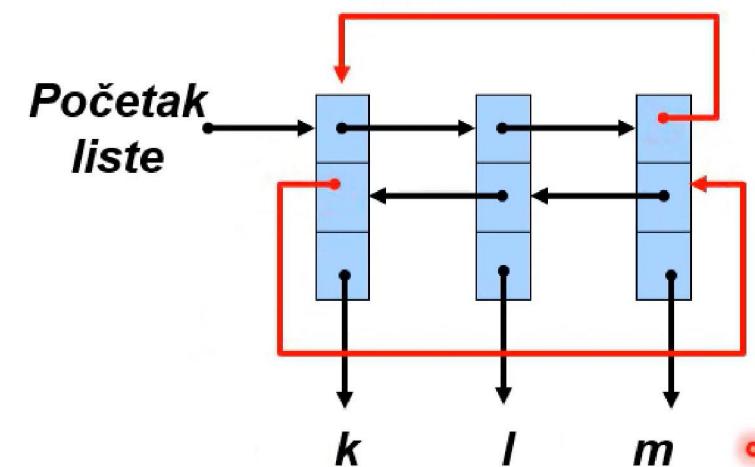
Jednostruko povezana linearna lista



Jednostruko povezana kružna lista



Dvostruko povezana linearna lista

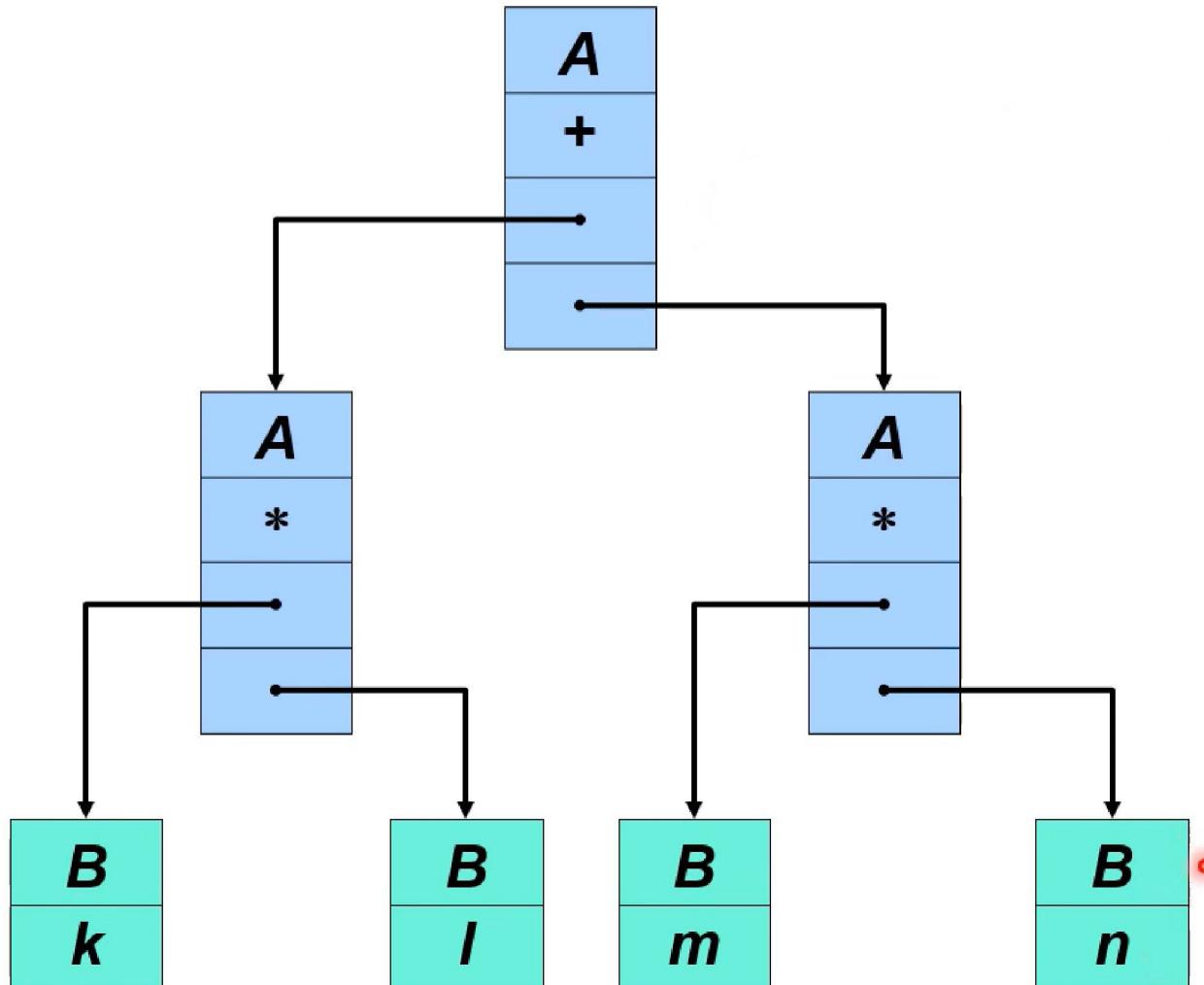


Dvostruko povezana kružna lista

- **Mreža**

- skup elemenata koji imaju različit broj polja
- polja sadrže
  - korisničke podatke
  - kazaljke koje pokazuju na druge elemente
- elementima se dodjeljuju
  - različiti tipovi
  - tipovi određuju ulogu pojedinog polja

# Podatkovni objekti ciljnog jezika



## Načini ostvarivanja apstraktnih tipova podataka

- **Apstraktne tipove podataka izvornog jezika**
  - potrebno je ostvariti podatkovnim objektima ciljnog jezika
- **Naredbe potpore izvođenju ciljnog programa**
  1. dodjeljuju memoriju apstraktnim podacima
  2. određuju način dohvata stavki apstraktnih podataka
  3. uspostavljaju mehanizme pristupa objektima ciljnog jezika



## Načini ostvarivanja apstraktnih tipova podataka

- **Jednostavni podaci**

- **Cijeli brojevi, prošireni cijeli brojevi i brojevi s posmačnim zarezom**
- **Izravno se ostvaruju odgovarajućim podatkovnim objektima procesora računala**
- **Veličina potrebne memorije**
  - veličina podatka
  - mogućnosti adresiranja procesora
- **Veličina podatka**
  - spremanje i dohvati isključivo s određene adrese
  - **Primjer**
    - *oktete* je moguće dohvatiti s bilo koje adrese
    - *riječi* i *duge riječi* moguće dohvatiti isključivo s adrese koja u binarnom zapisu završava određenim brojem nula



## Načini ostvarivanja apstraktnih tipova podataka

- **Niz**
  - **ostvaruje se vektorom**
  - **niz promjenjive veličine, niz se dijeli i nadovezuje**
    - **ostvaruje se listom**
    - **zauzima više memorijskog prostora**
    - **omogućava brži rad**



## Načini ostvarivanja apstraktnih tipova podataka

- **Stog**

- moguće je unaprijed odrediti maksimalnu veličinu stoga
  - ostvaruje se vektorom
  - veličina elementa vektora
    - veličina stavke stoga
  - veličina vektora
    - maksimalna veličina stoga
  - adresa početnog elementa vektora
    - mjesto zapisa prve podatkovne stavke stoga
  - dohvati i dodavanje stavke
    - kazaljka koja pokazuje na element vektora koji je posljednji spremlijen u memoriju
  - učinkovit dohvat elementa
    - indeksirani način adresiranja s naknadnim ili prethodnim inkrementiranjem i dekrementiranjem
- nije moguće unaprijed odrediti maksimalnu veličinu
  - ostvaruje se listom
  - rast stoga
    - dinamička dodjela elemenata liste



## Načini ostvarivanja apstraktnih tipova podataka

- **Red**
  - ostvaruje se vektorom
  - FIFO načina adresiranja procesora računala
  - nedostatak
    - pomak reda u memorijskom prostoru tijekom dohvata stavki s početka reda i dodavanja stavki na kraj reda
  - **kružna lista**
    - kazaljke koja pokazuje na početak i kraj kružne liste omogućuju brzi dohvat i dodavanje stavki
- **Stablo i usmjereni graf**
  - ostvaruju se listama i mrežama
- **Tablice**
  - ostvaruju se vektorima i mrežama



# Načini ostvarivanja apstraktnih tipova podataka

- **Polja**

- **ostvaruju se vektorima**
- **jednodimenzionalno polje *Polje[i:j]***
  - ostvaruje se vektorom *Vektor[1:j-i+1]*
  - stavka *Polje[k]* ostvaruje se elementom *Vektor[k-i+1]*
- **višedimenzionalna polja**
  - ostvaruju se vektorima
- **dvodimenzionalno polje**
  - ostvaruje se elementima vektora
    - redak po redak
    - stupac po stupac
  - **Primjer**
  - **polje *A[1:2,3:5]***
    - *V[1]=A[1,3], V[2]=A[1,4], V[3]=A[1,5], V[4]=A[2,3], V[5]=A[2,4], V[6]=A[2,5]*
    - *V[1]=A[1,3], V[2]=A[2,3], V[3]=A[1,4], V[4]=A[2,4], V[5]=A[1,5], V[6]=A[2,5]*



## Načini ostvarivanja apstraktnih tipova podataka

- **Polje**  $\text{Polje}[i_1:k_1, i_2:k_2, i_3:k_3, \dots, i_n:k_n]$ 
  - stavka  $\text{Polje}[j_1:j_2:j_3, \dots, j_n]$  ostvaruje se sljedećim elementom:

$$\text{Vektor}\left[1 + \sum_{m=1}^n (j_m - i_m) D_m\right]$$

- **Vrijednosti**
  - $D_1 = 1$
  - $D_m = (k_{m-1} - i_{m-1} + 1)D_{m-1}$
  - svaki dohvati stavke
    - potrebno je izračunati zadane izraze
- **Dijelove izraza koji ne ovise o indeksima**
  - moguće je unaprijed izračunati i spremiti

$$C_n = \sum_{m=1}^n i_m D_m$$



## Načini ostvarivanja apstraktnih tipova podataka

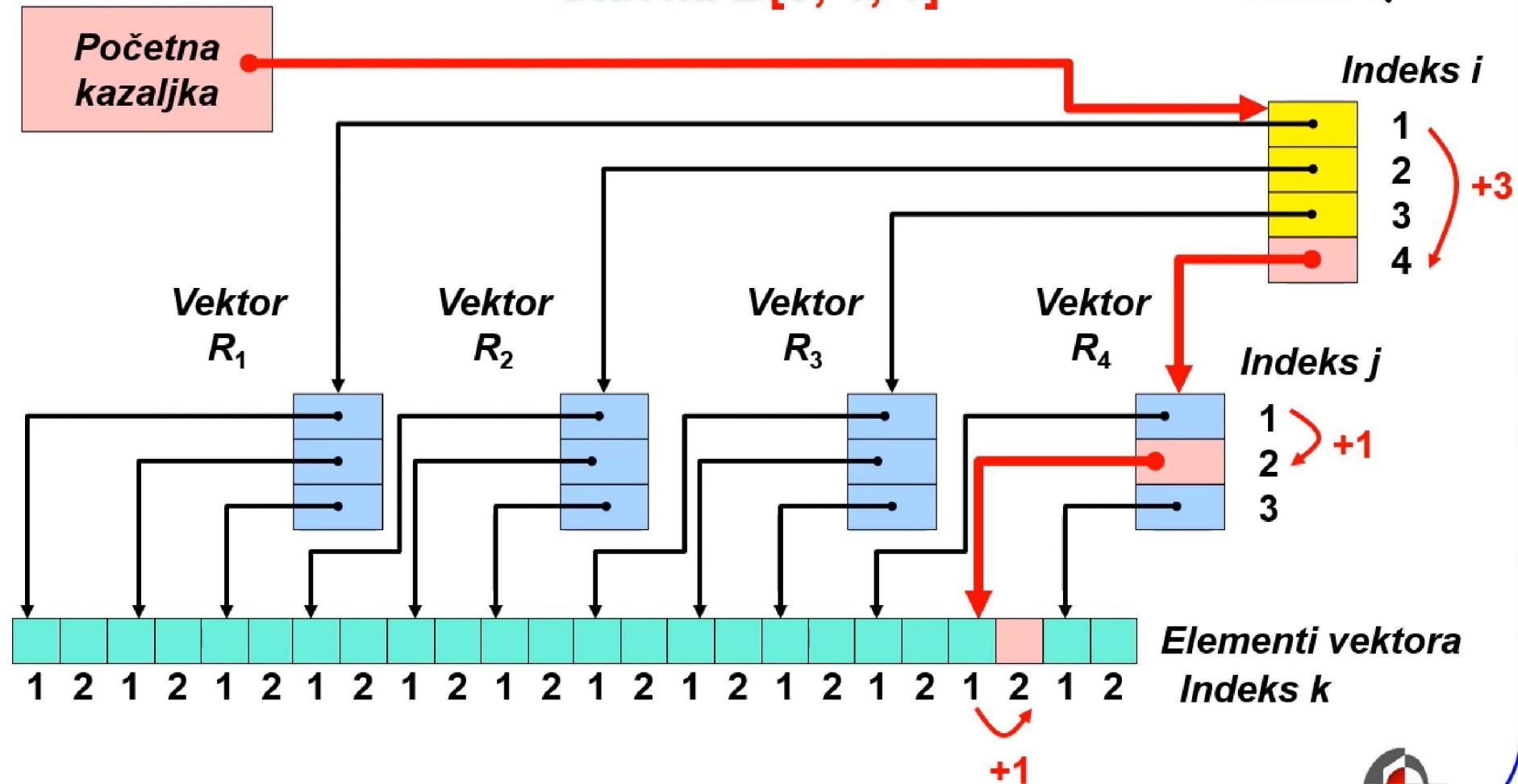
- **Više istovjetnih polja**
  - jednom izračunate vrijednosti koriste se za sva polja
- **U memoriju računala spremi se**
  - dimenzija polja
  - gornja i donja granica za sve indekse polja
  - broj stavki u polju
  - vrijednosti izraza  $D_m$
  - vrijednosti izraza  $C_n$
- **Predloženo rješenje**
  - zahtijeva množenje tijekom svakog dohvata stavke
  - značajno usporava izvođenje programa
- **Izostanak množenja**
  - uporaba hijerarhije kazaljki indeksa



# Načini ostvarivanja apstraktnih tipova podataka

$i \quad j \quad k$   
**Polje  $B[0:3, 0:2, 0:1]$**

**Stavka  $B[3, 1, 1]$**



## Tijek izvođenja izvornog programa zasnovanog na procedurama

- **Proceduralni jezici**

- C, FORTRAN, Java, itd.
- **Tijek izvođenja proceduralnih jezika**
  1. Izvorni program izvodi se slijedno onim redoslijedom kojim su naredbe zapisane u tekstu izvornog programa
  2. U izvornom programu posebice se ističe jedna naredba koja započinje njegovo izvođenje
  3. Izvođenje procedure prekida slijedno izvođenje izvornog programa
  4. Nakon poziva procedure, izvođenje izvornog programa nastavlja se njezinom početnom naredbom
  5. Naredbe procedure izvode se slijedno sve do završne naredbe
  6. Nakon izvedene završne naredbe procedure nastavlja se izvoditi prva naredba koja slijedi naredbu poziva procedure
- **Životni vijekovi više procedura**
  - Životni vijek jedne procedure završi prije nego što započne životni vijek druge procedure
  - Životni vijekovi su ugniježđeni



## Započinje *Glavni()*

Započinje *C(2,5)*

Započinje *C(3,4)*

*Glavni()*

*C(2,5)*

*A(2)*

*C(3,5)*

*B(5)*

*C(3,4)*

*A(3)*

*C(4,5)*

*B(5)*

*C(4,4)*

*A(3)*

*C(4,4)*

*B(4)*

*C(4,3)*

*A(4)*

*C(5,5)*

*B(5)*

*C(5,4)*

Započinje *C(3,4)*

Završava *C(2,5)*

Završava *Glavni()*

*Glavni()*

*A(x)*

{

*x = x+1;*  
*vrati (x);*

}

*B(y)*

{

*y = y-1;*  
*vrati (y);*

}

*C(k,m)*

{

*ako (k < m)*  
{  
*k = A(k);*  
*C(k,m);*  
*m = B(m);*  
*C(k,m);*  
}

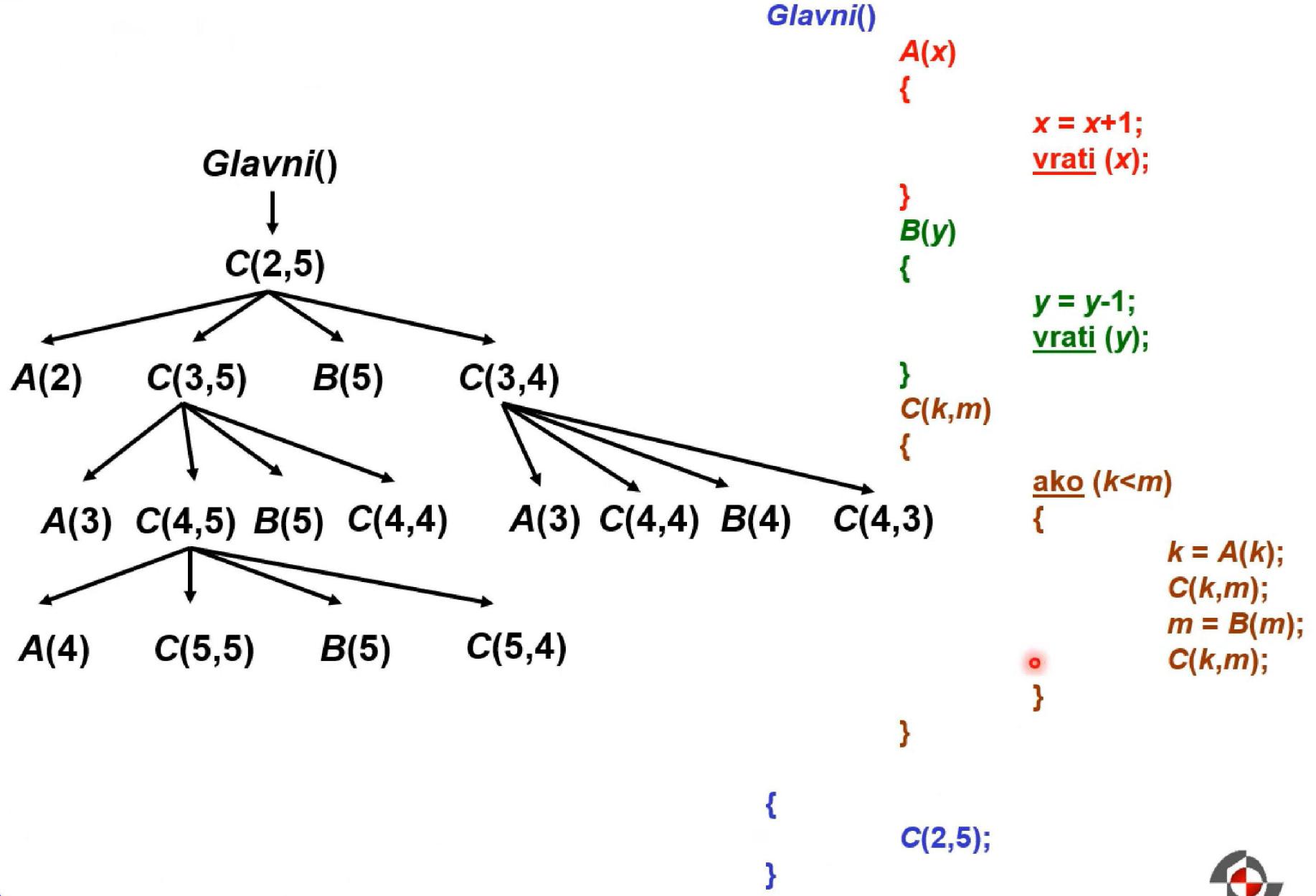
}

{

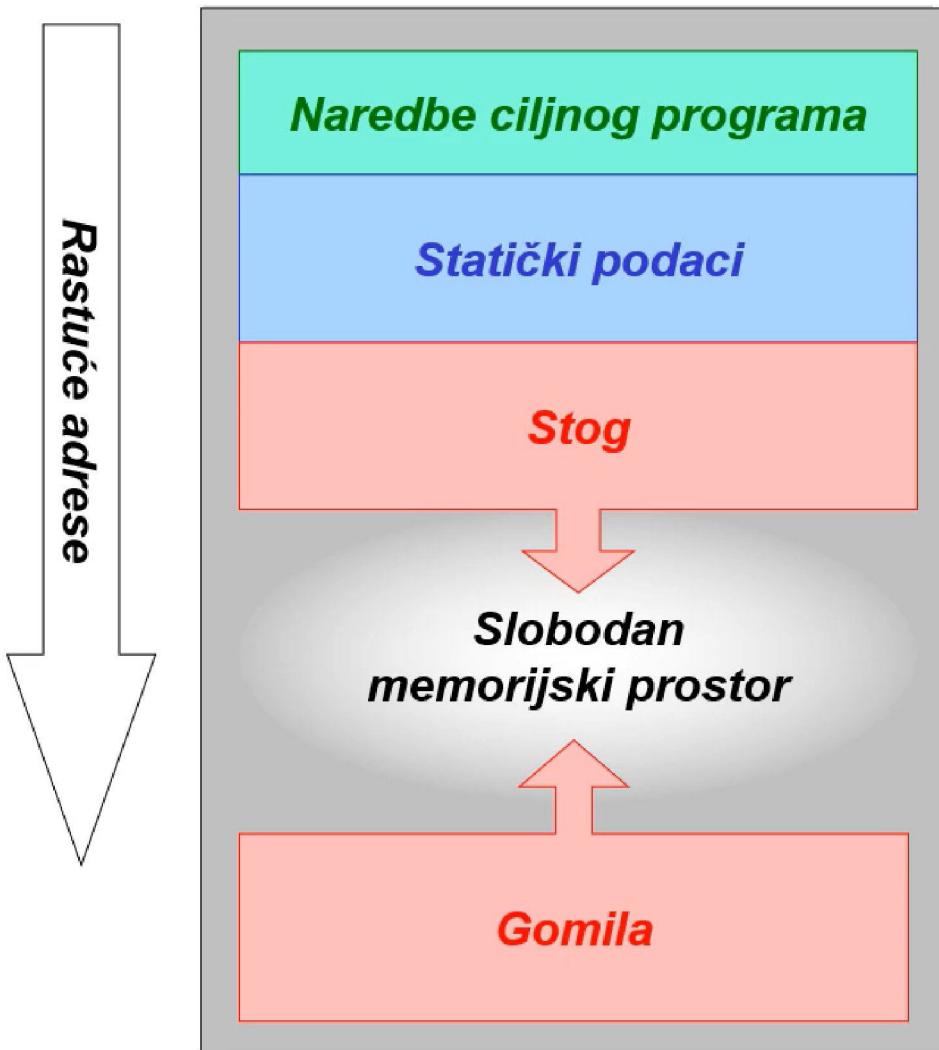
}

*C(2,5);*





# Organizacija i postupci dodjele memorije



- Dodjela statičke memorije
- Dodjela memorije na stogu
- Dodjela memorije gomile

# Dodjela statičke memorije

- **Imena izvornog programa**
  - pridruživanje podatkovnih objekata ciljnog programa
    - tijekom prevođenja izvornog u ciljni program
  - nema potrebe za dodatnim naredbama potpore izvođenju ciljnog programa
  - FORTRAN
- **Statička memorija**
  - naredbe ciljnog programa
  - podaci
    - moguće je odrediti njihovu veličinu tijekom prevođenja izvornog programa u ciljni program
    - veličina se ne mijenja tijekom izvođenja ciljnog programa



# Dodjela statičke memorije

- **Prednost**
  - mogućnost određivanja memorijskih adresa tijekom prevodenja izvornog programa u ciljni program
  - **dostupna** adresa početka statičke memorije
    - u naredbe ciljnog programa ugrade se **potpuno izračunate** adrese
  - **nedostupna** adresa početka statičke memorije
    - u naredbe se ugrađuju **djelomično izračunate** adrese

# Dodjela statičke memorije

- **Djelomično izračunate adrese**
  - pomaci od početne adrese statičke memorije
  - tijekom punjenja ciljnog programa u dodijeljeni dio statičke memorije
    - relociranje ili premještanje
      - prethodno izračunatim pomacima doda se vrijednost početne adrese statičke memorije
      - postupak premještanja je dio pripreme ciljnog programa za izvođenje

# Dodjela statičke memorije

- **Opisnik procedure u statičkoj memoriji**
  - **nakon izvođenja procedure**
    - ostaju sačuvane vrijednosti lokalnih podataka
  - **ponovno aktivirana procedura**
    - koristi prethodno spremljene vrijednosti lokalnih podataka



# Dodjela statičke memorije

- **Nedostaci uporabe statičke memorije**
  - potreba poznavanja veličine podatka tijekom prevodenja izvornog progama u ciljni program
  - nemogućnost dinamičke dodjele memorije
  - nemogućnost izvođenja rekurzivnih procedura
    - lokalno ime vezano je isključivo na jedan podatkovni objekt
    - nije moguće da rekurzivno aktivirana procedura koristi nove vrijednosti lokalnog podatka

# Dodjela memorije na stogu

- **Upravljački stog programskih jezika**
  - Spremanje opisnika procedura
  - Pascal i C
  - **Početak izvođenja procedure**
    1. prekida se izvođenje pozivajuće procedure
    2. opisnik se stavlja na vrh stoga
    3. popunjava se opisnik pozvane procedure
  - **Završetak izvođenja procedure**
    - podaci u opisniku omogućuju nastavak izvođenja prethodno prekinute pozivajuće procedure



# Dodjela memorije na stogu

- **Dodjela memorije na stogu**
  - uvećavanje ili smanjivanje vrijednosti kazaljke stoga za veličinu opisnika pozvane procedure
  - stara vrijednost kazaljke stoga spremi se u opisnik pozvane procedure
- **Dohvat podatka sa stoga**
  - adresi kazaljke stoga doda se ili oduzme odgovarajući pomak
  - za dohvatz podatka koristi se
    - izravno registar kazaljke stoga procesora
    - adresni registar u koji se prethodno spremi vrijednost kazaljke stoga
      - adresiranje s pomakom

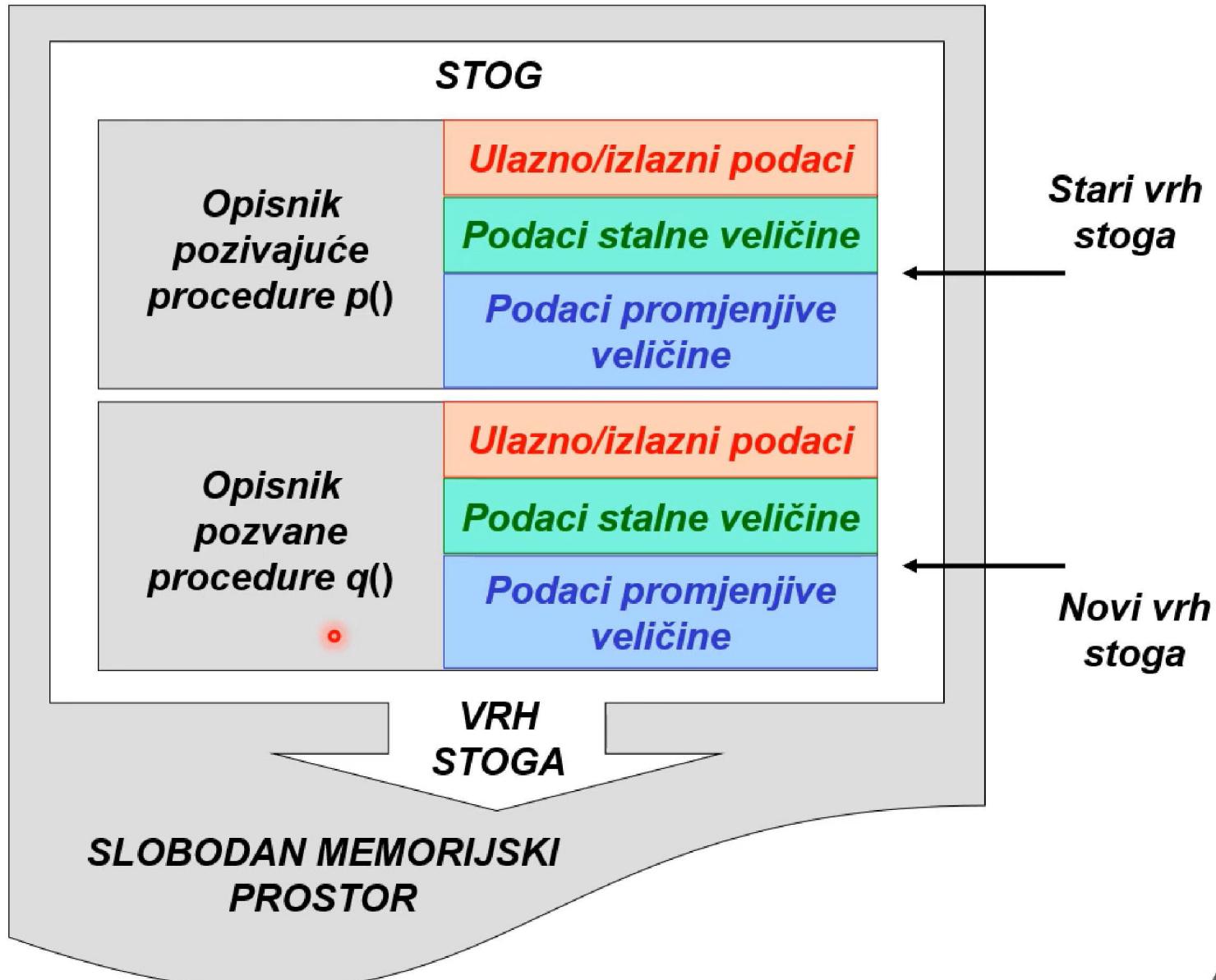


# Dodjela memorije na stogu

- **Izuzimanje opisnika**
  - ostvaruje se čitanjem prethodno spremljene stare vrijednosti kazaljke stoga
    - oslobađa se memorija na stogu
- **Posebne strojne naredbe**
  - **Dodjela memorije na stogu**
    - Naredba omogućuje
      1. brzo i jednostavno spremanje na stog stare vrijednosti kazaljke stoga
      2. pridruživanje nove vrijednosti kazaljci stoga
    - **Oslobađanje memorije na stogu**
      - Naredba omogućuje
        - dohvat sa stoga prethodno spremljene stare vrijednosti kazaljke stoga



# Dodjela memorije na stogu

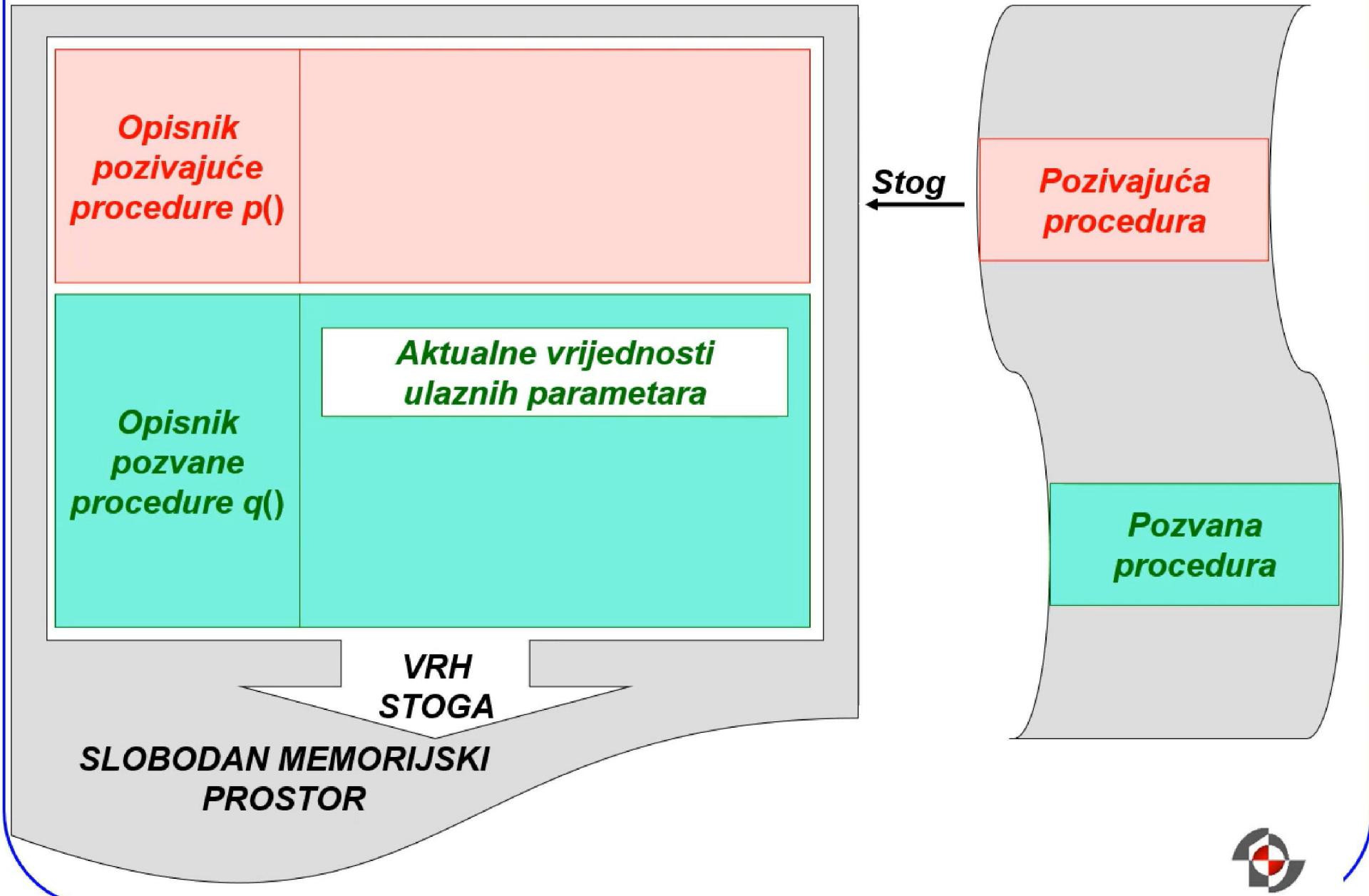


# Dodjela memorije na stogu

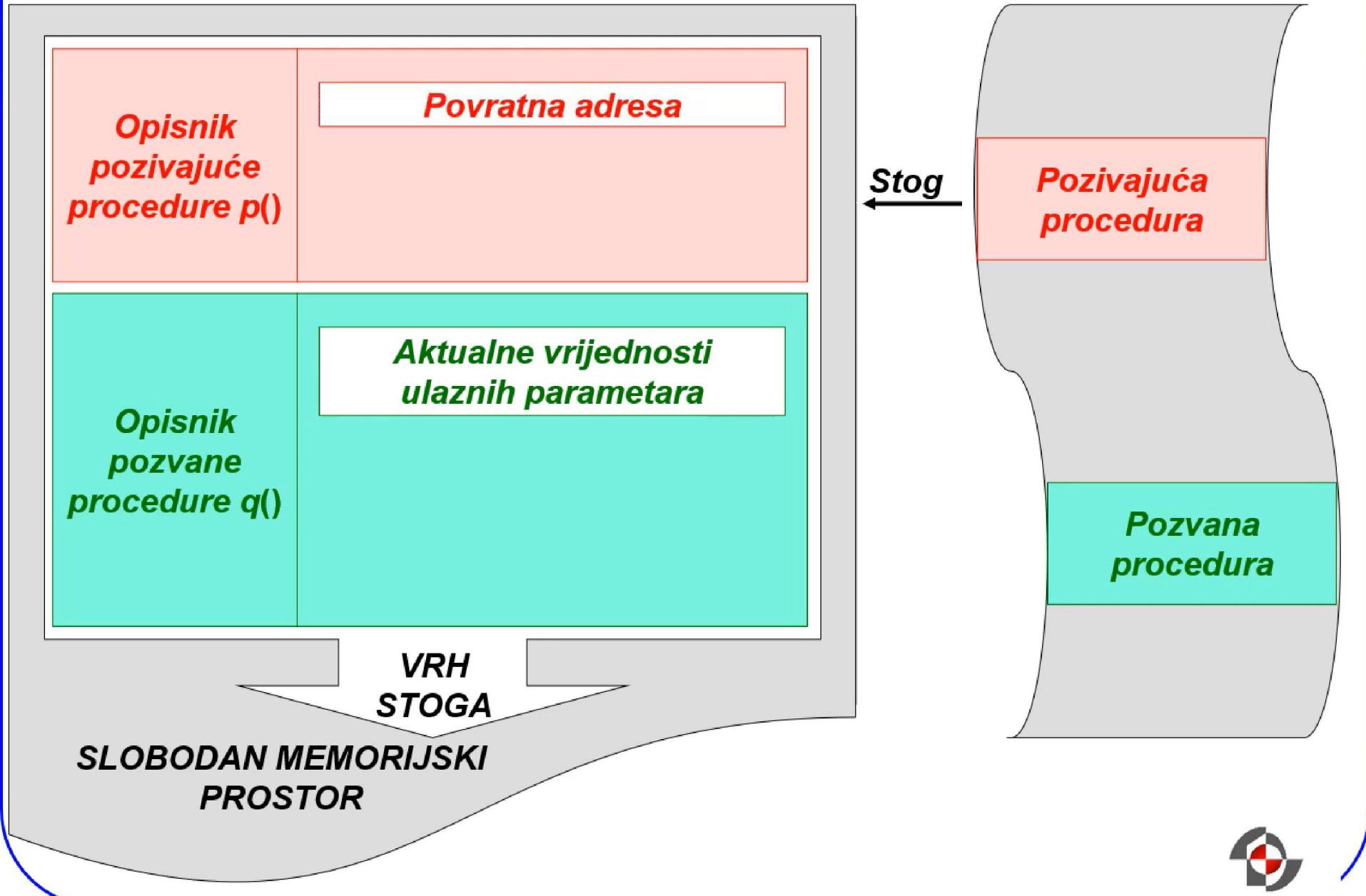
- **Naredbe potpore izvođenju ciljnog programa**
  - dinamički upravljaju opisnicima na stogu
  - popunjavaju opisnike odgovarajućim podacima
  - naredbe dinamičkog upravljanja opisnicima izvodi
    - pozivajuća procedura
    - pozvana procedura



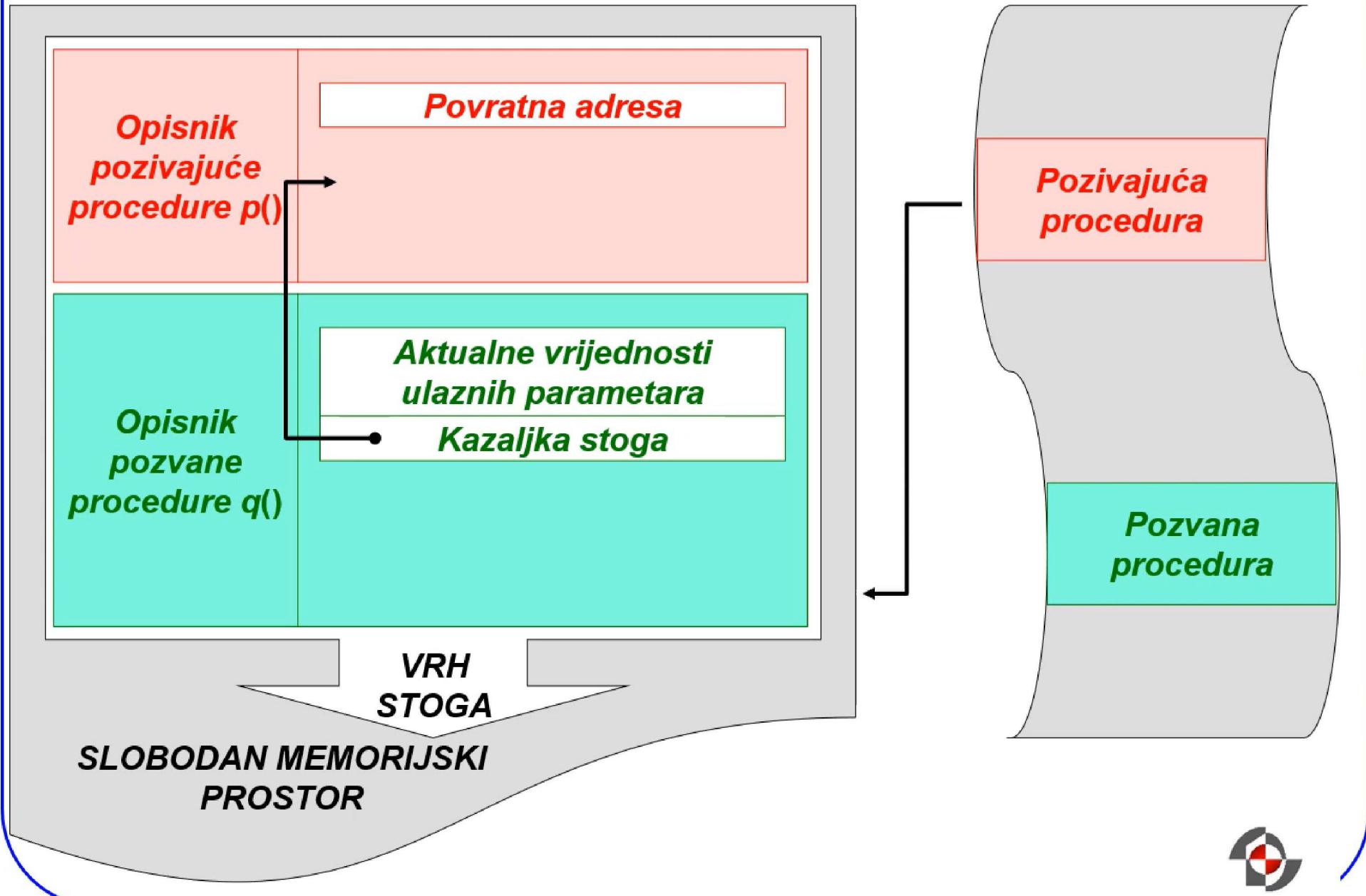
# Dodjela memorije na stogu



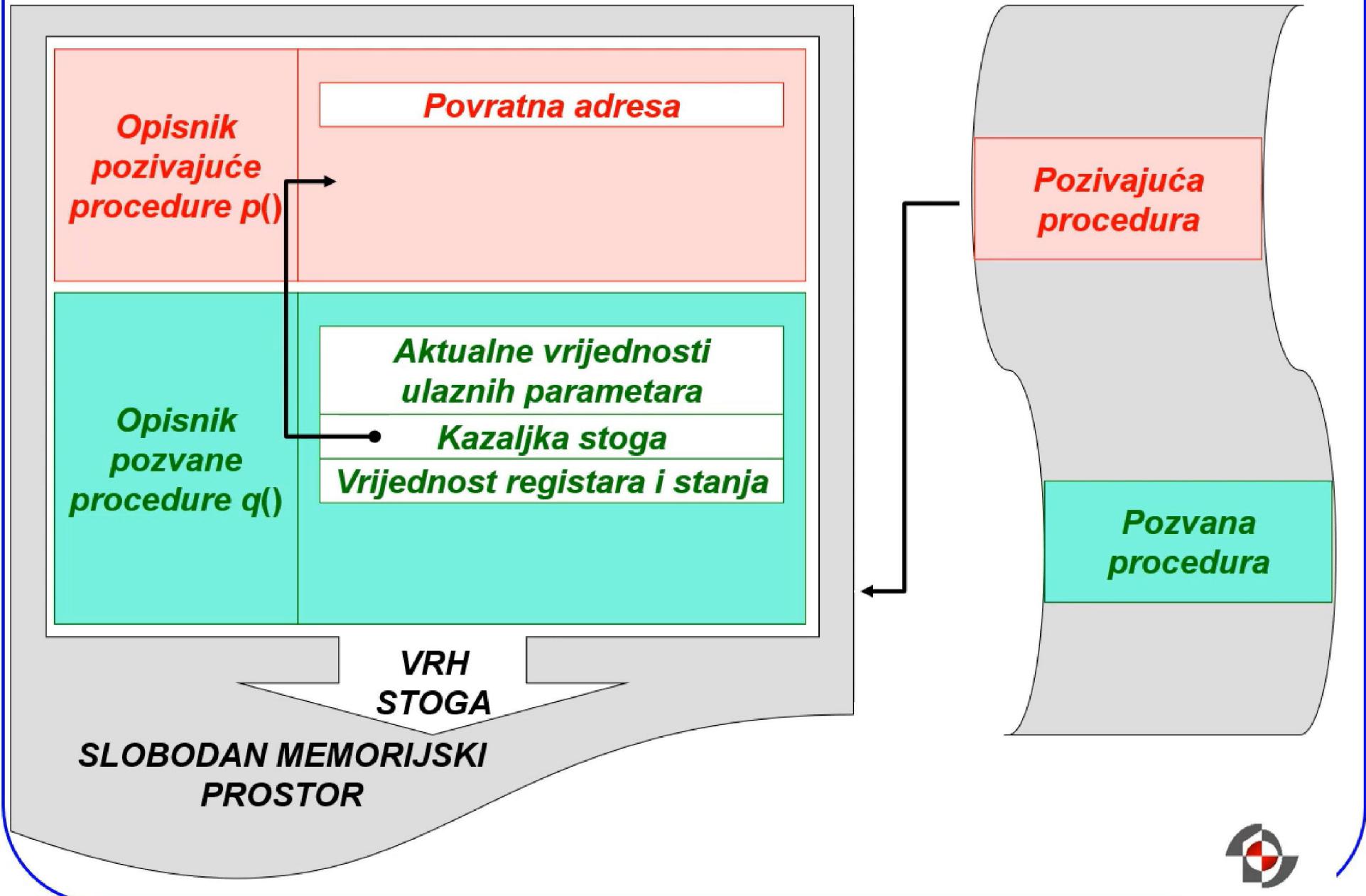
# Dodjela memorije na stogu



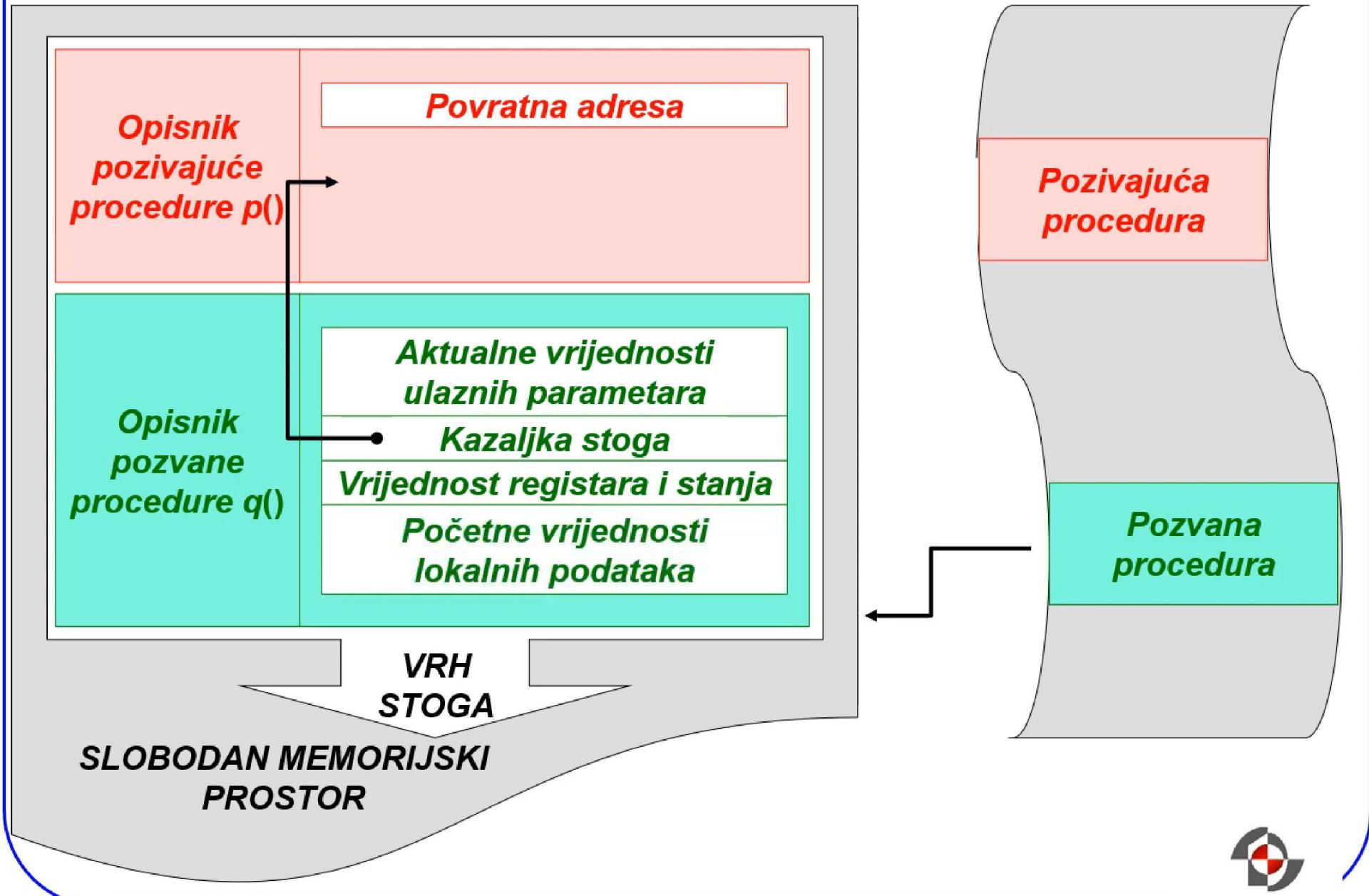
# Dodjela memorije na stogu



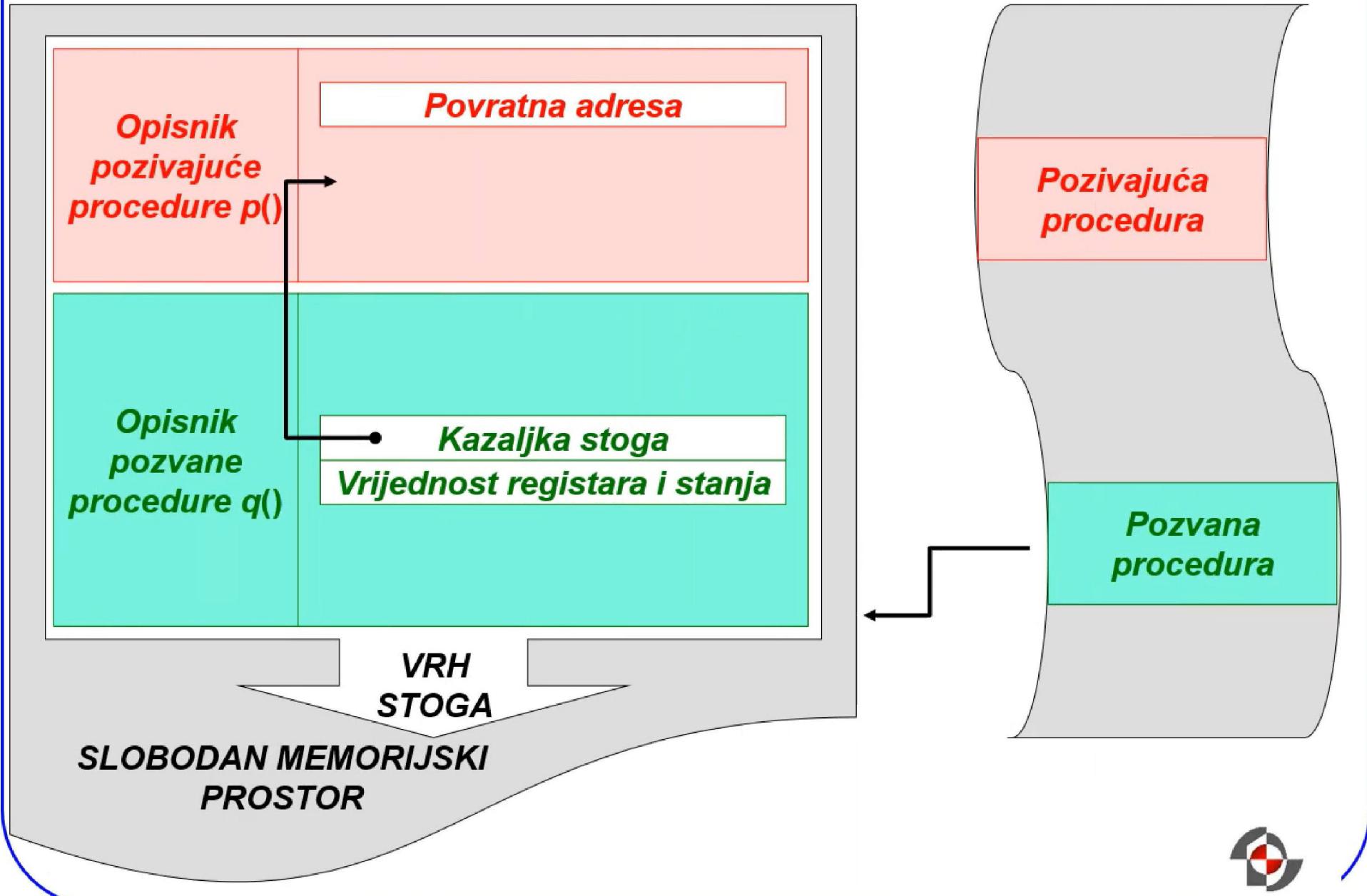
# Dodjela memorije na stogu



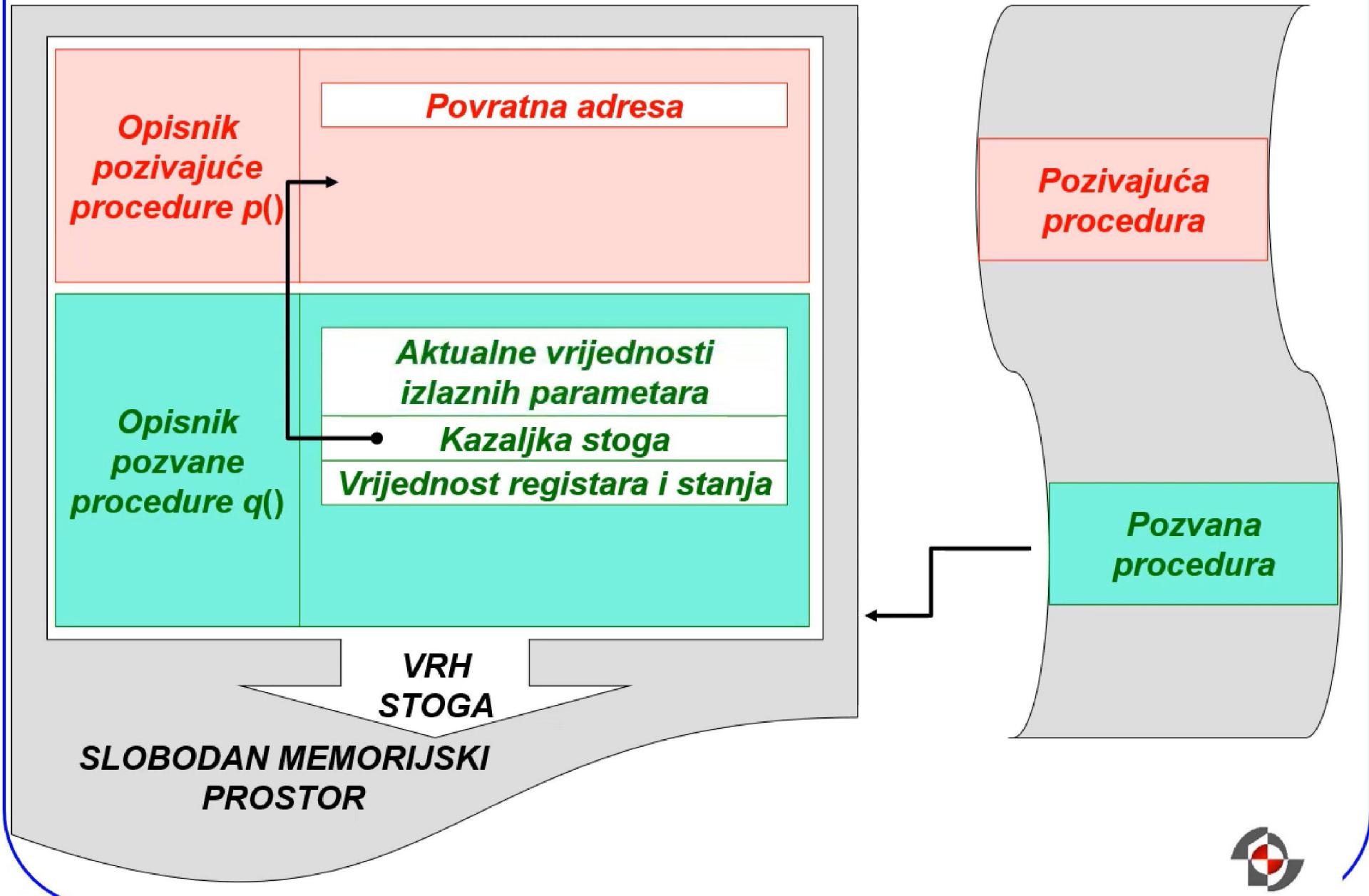
# Dodjela memorije na stogu



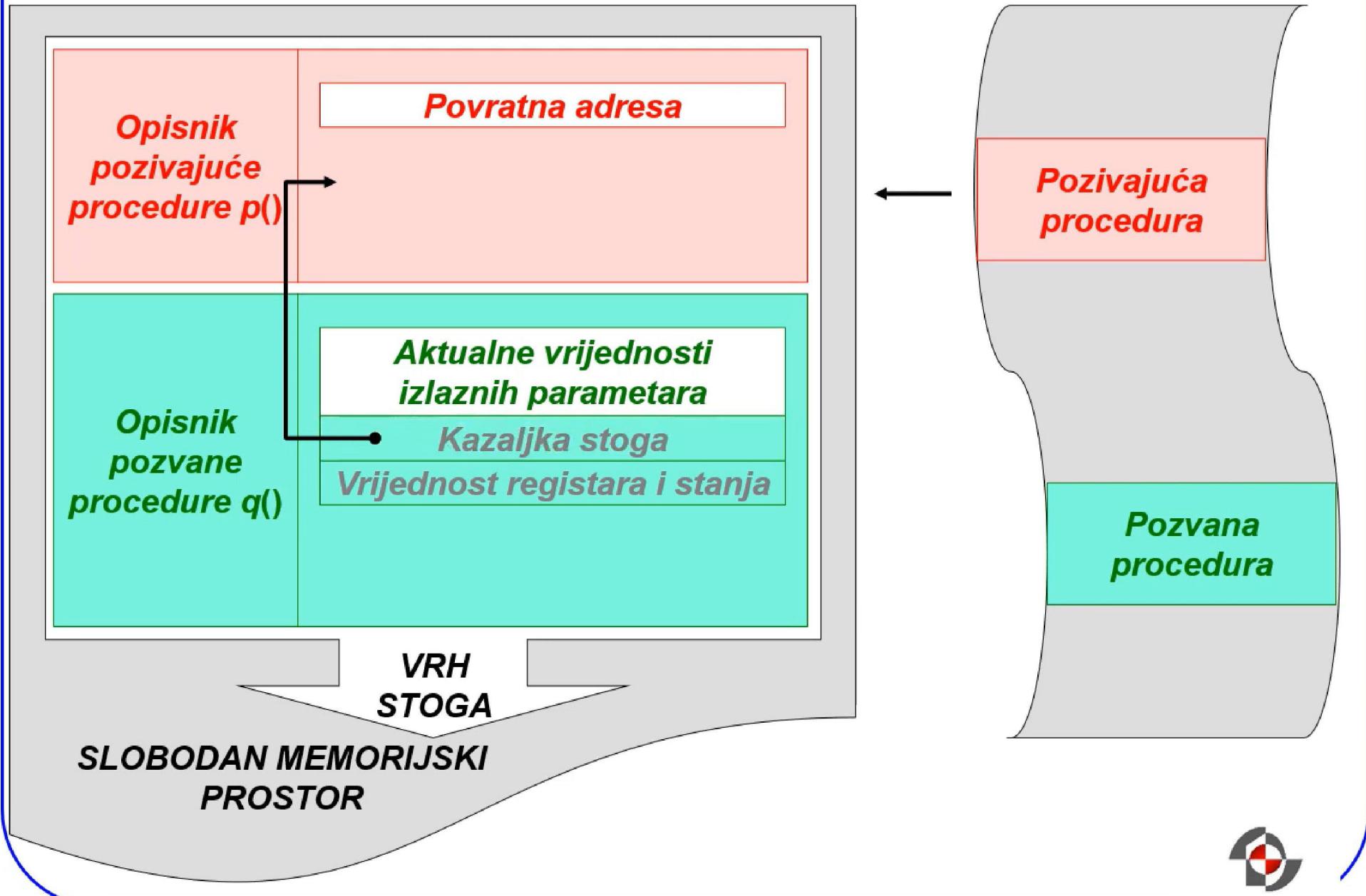
# Dodjela memorije na stogu



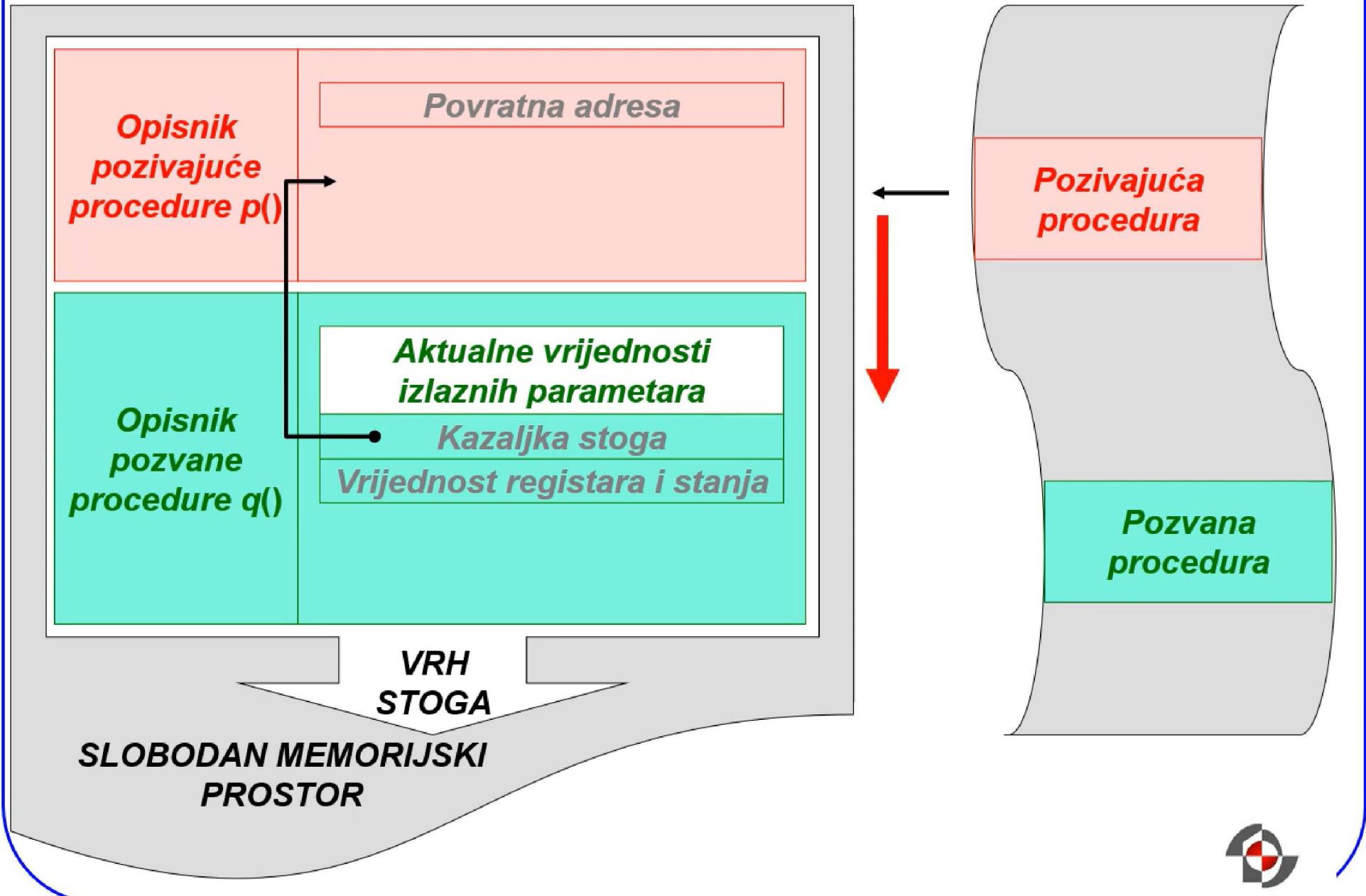
# Dodjela memorije na stogu



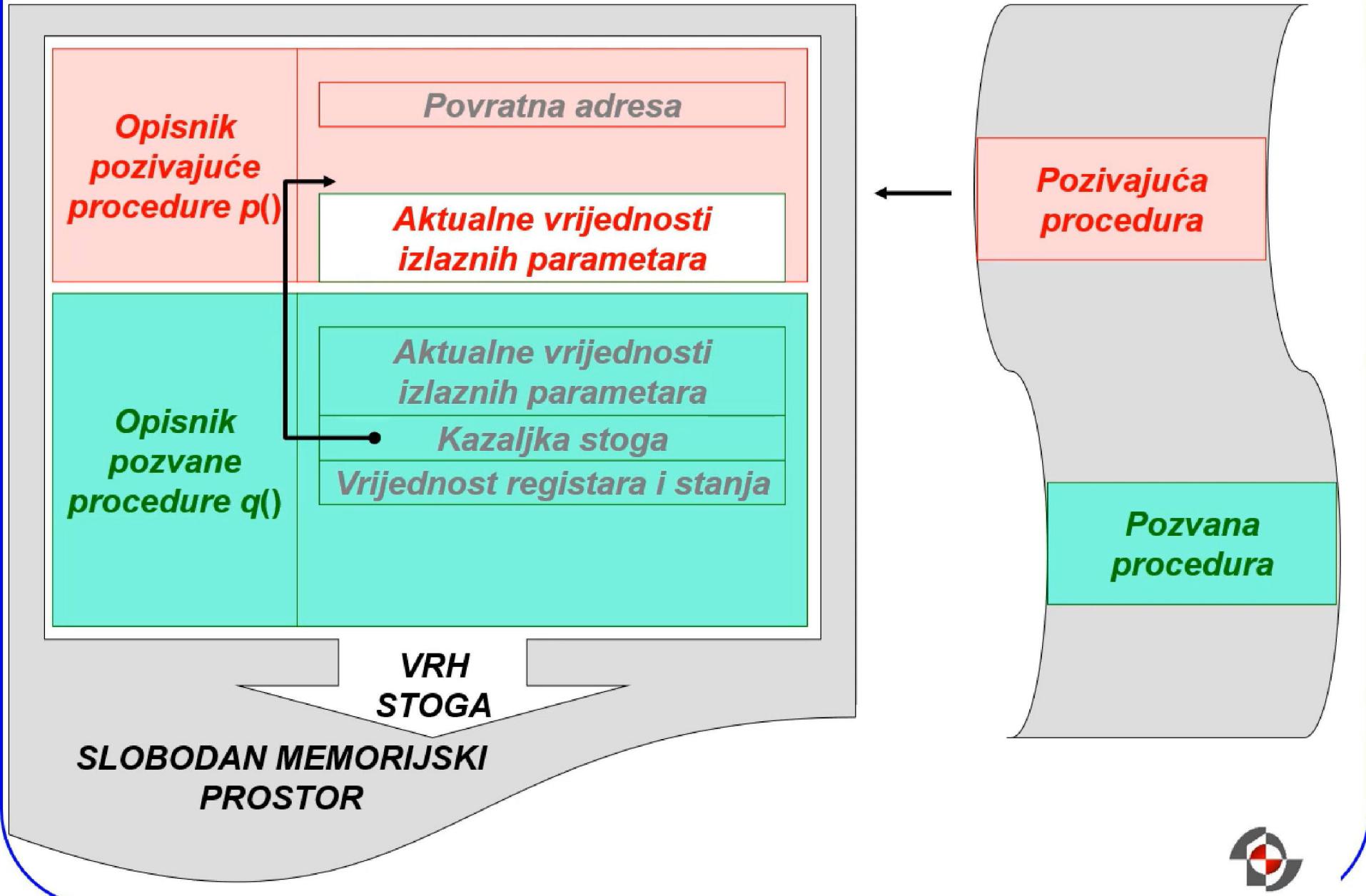
# Dodjela memorije na stogu

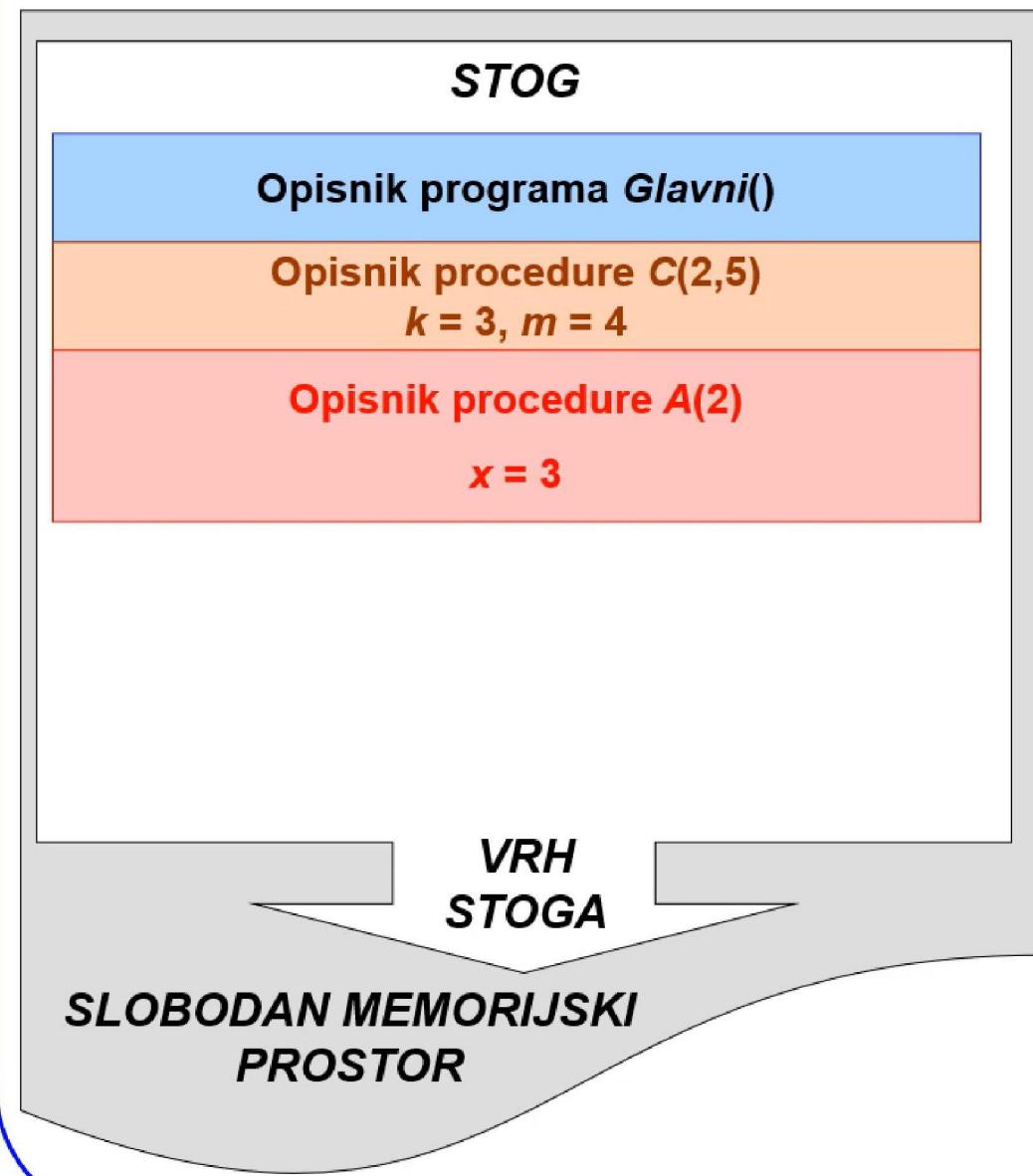


# Dodjela memorije na stogu



# Dodjela memorije na stogu





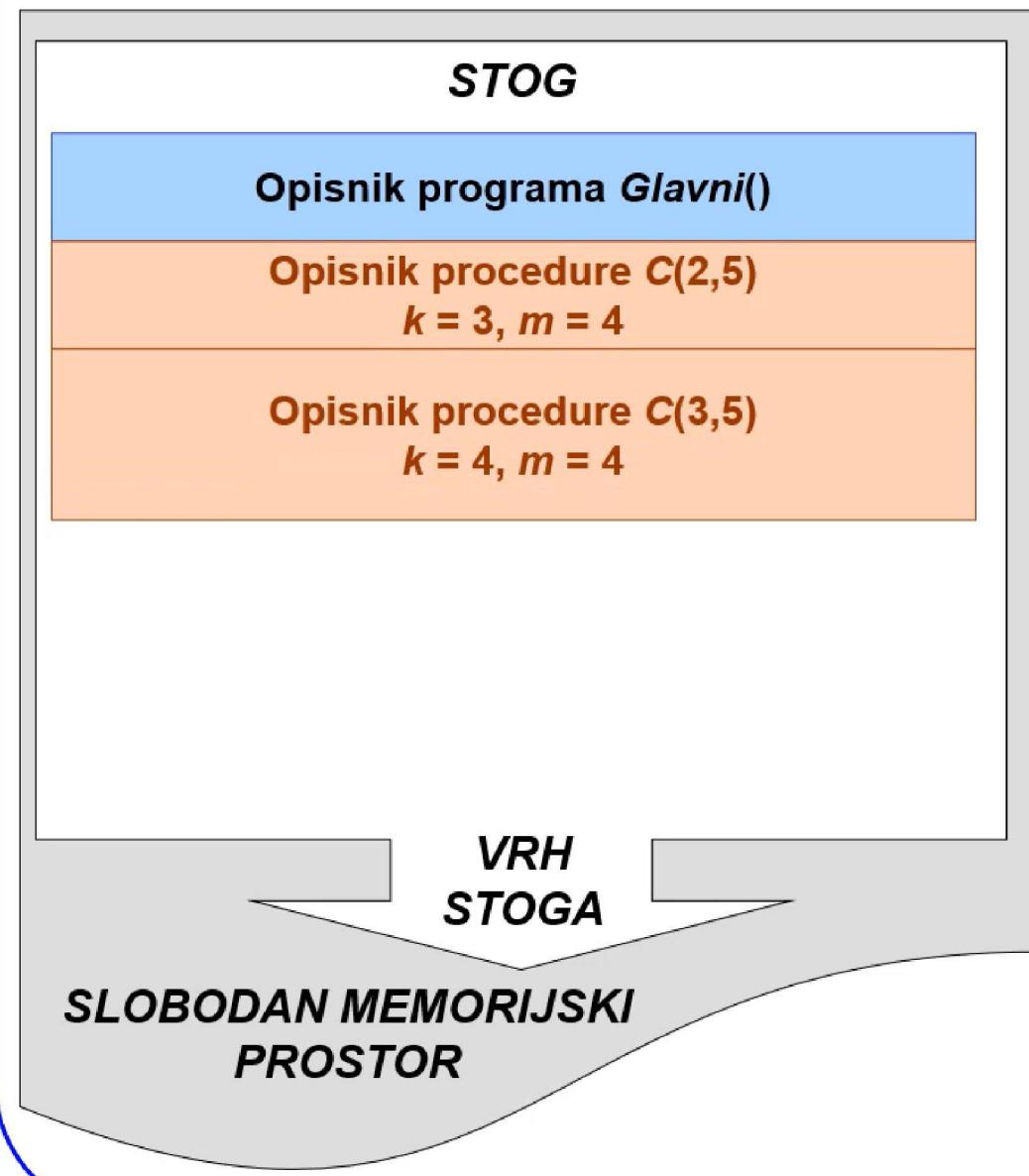
*Glavni()*

```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```



*Glavni()*

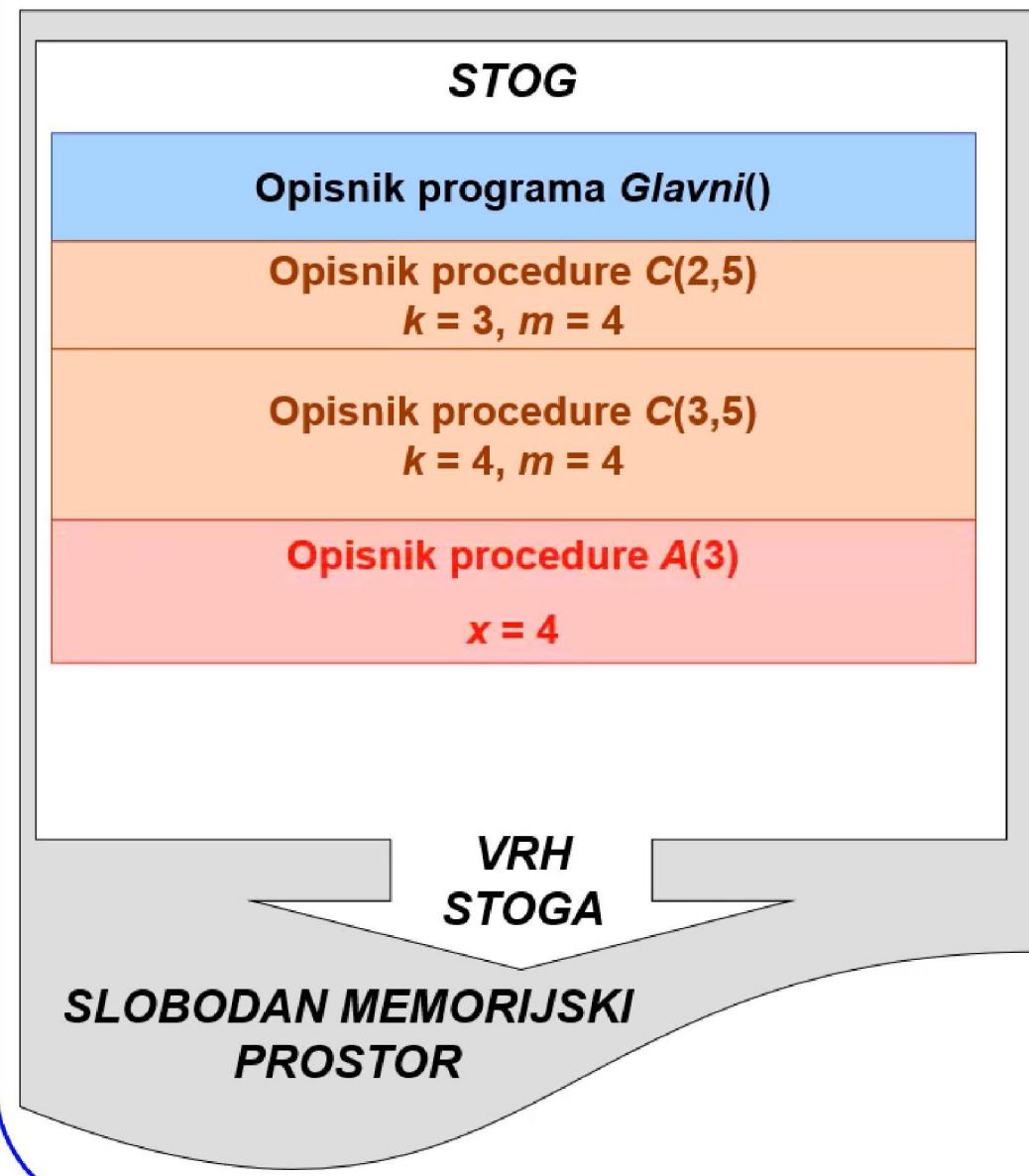
```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```





*Glavni()*

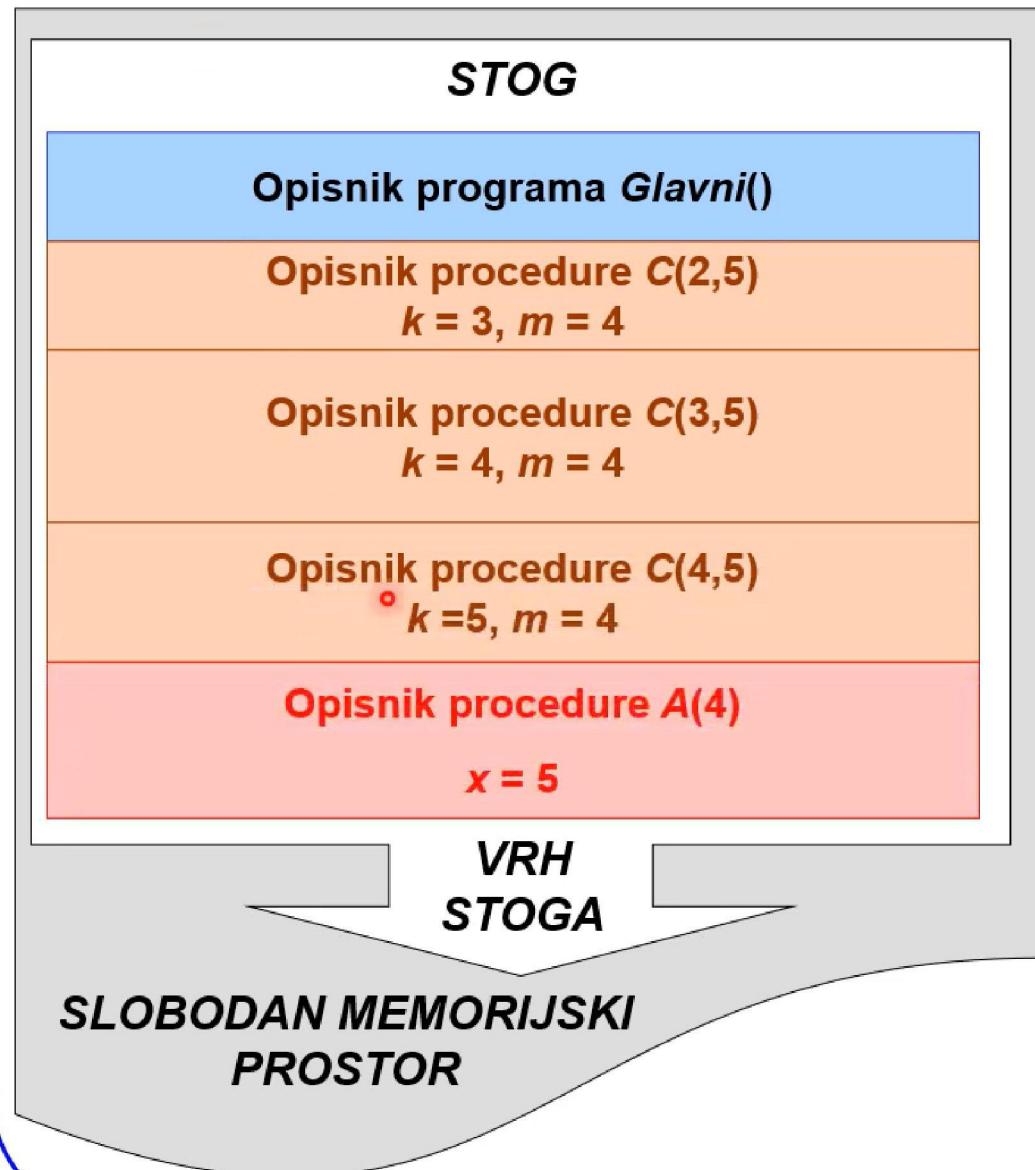
```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```





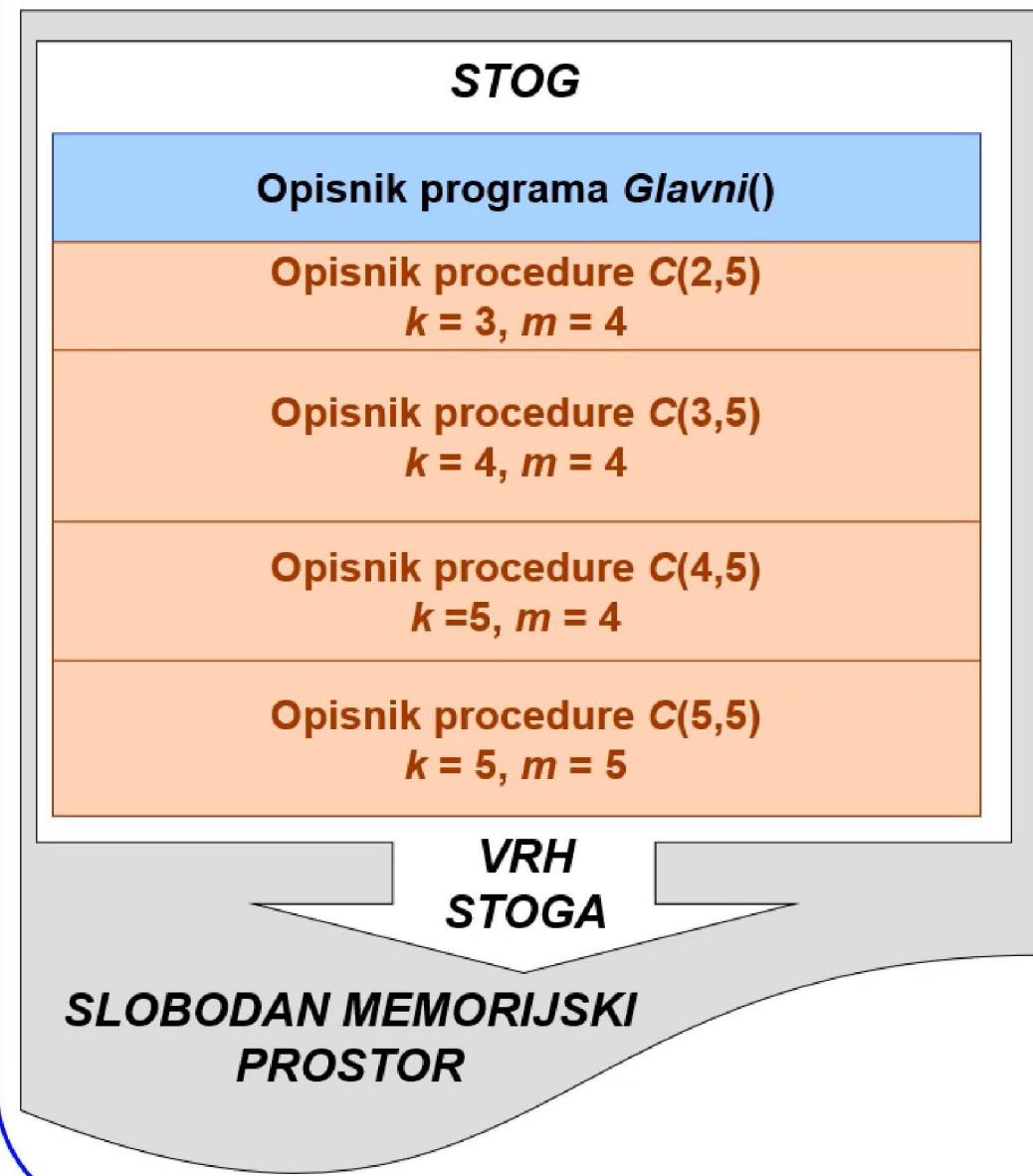
*Glavni()*

```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```



*Glavni()*

*A(x)*  
{

$x = x+1;$   
vrati (x);

}

*B(y)*  
{

$y = y-1;$   
vrati (y);

}

*C(k,m)*  
{

ako ( $k < m$ )  
{

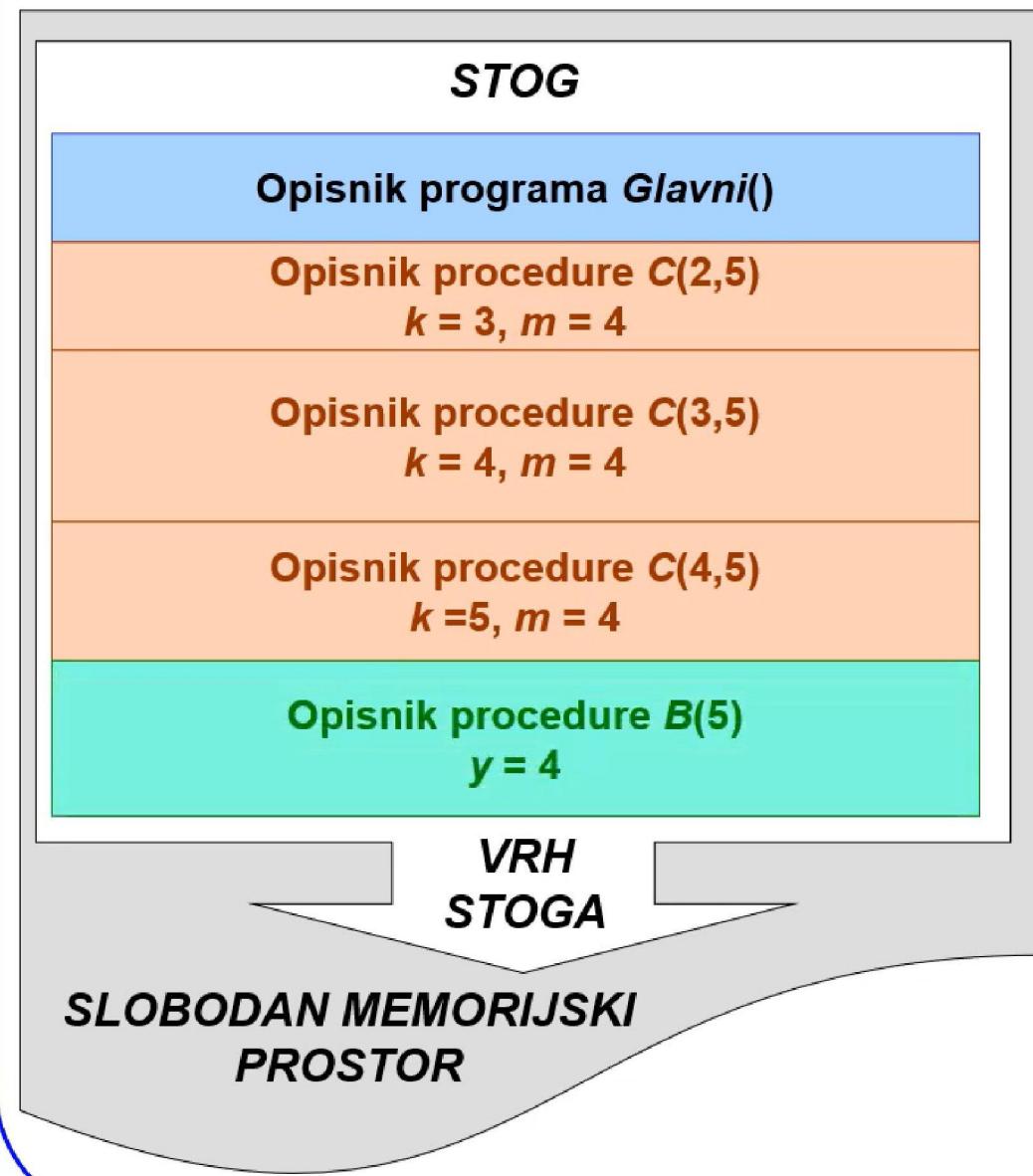
$k = A(k);$   
*C(k,m);*  
 $m = B(m);$   
*C(k,m);*

}

{  
}

*C(2,5);*





*Glavni()*

*A(x)*  
{

$x = x+1;$   
vrati (x);

}

*B(y)*  
{

$y = y-1;$   
vrati (y);

}

*C(k,m)*  
{

ako ( $k < m$ )  
{

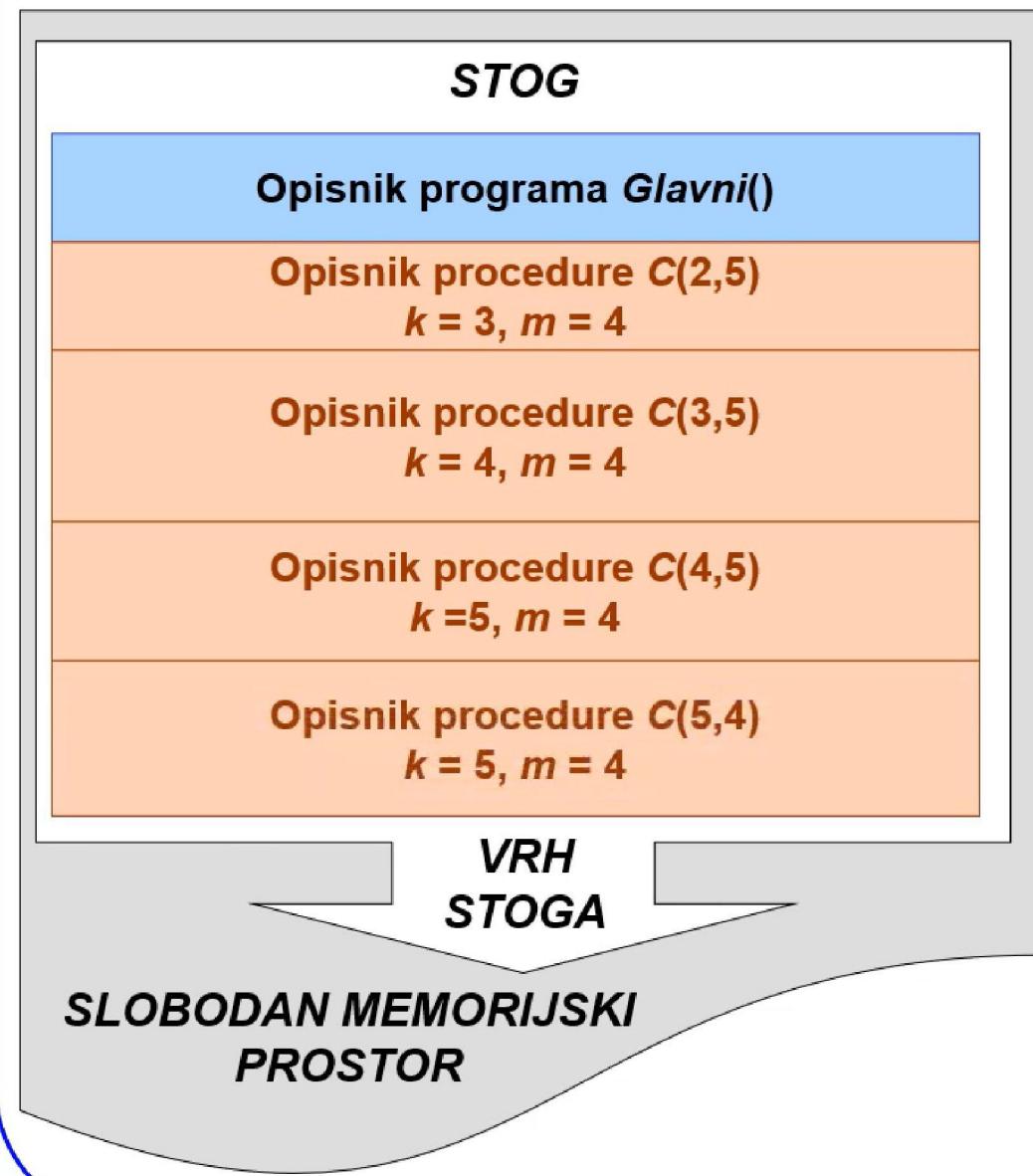
$k = A(k);$   
*C(k,m);*  
 $m = B(m);$   
*C(k,m);*

}

{  
}

*C(2,5);*





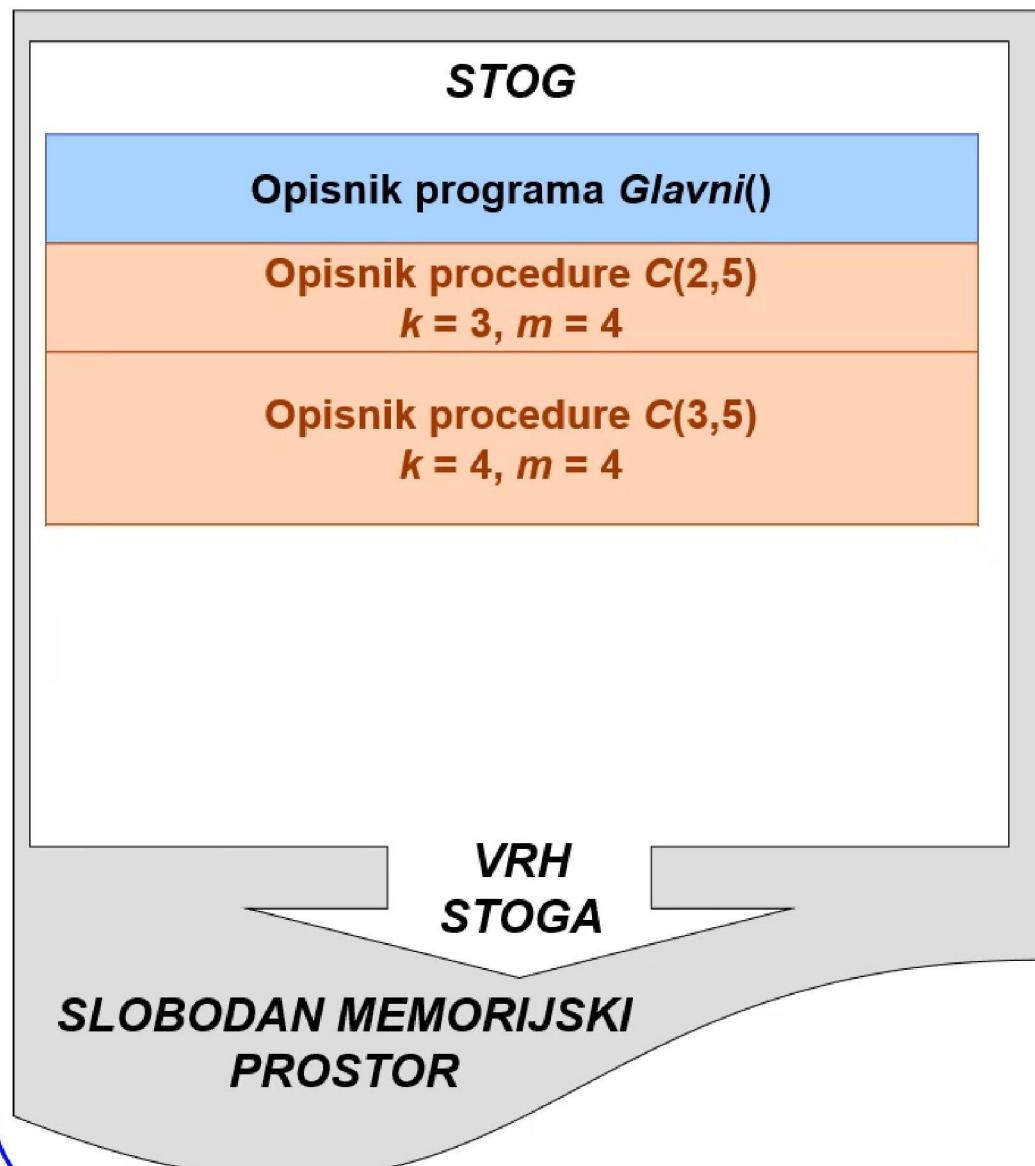
*Glavni()*

```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```



*Glavni()*

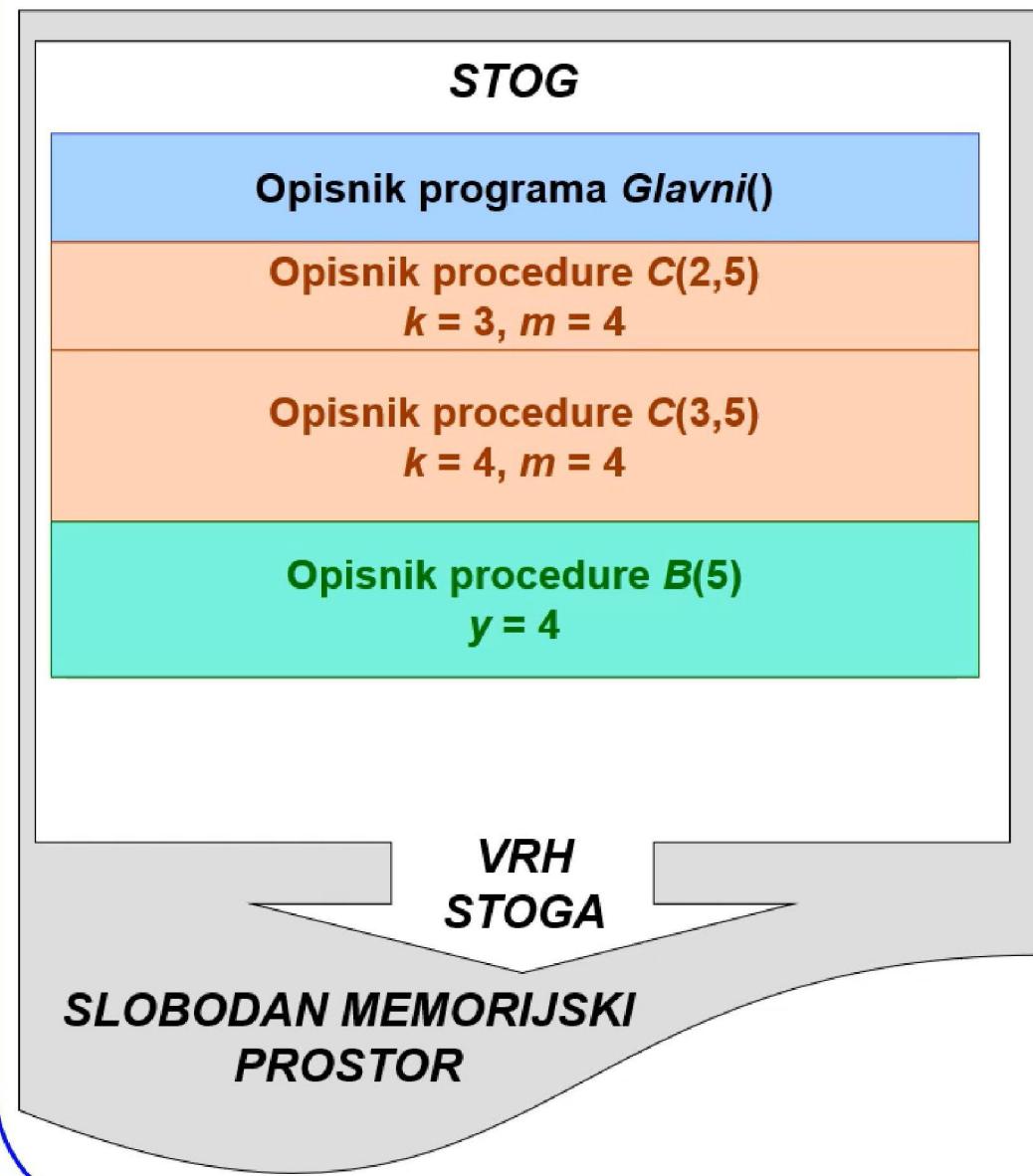
```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```





*Glavni()*

*A(x)*  
{

*x* = *x*+1;  
vrati (*x*);

}

*B(y)*  
{

*y* = *y*-1;  
vrati (*y*);

}

*C(k,m)*  
{

ako (*k*<*m*)  
{

*k* = *A(k)*;  
*C(k,m)*;  
*m* = *B(m)*;  
*C(k,m)*;

}

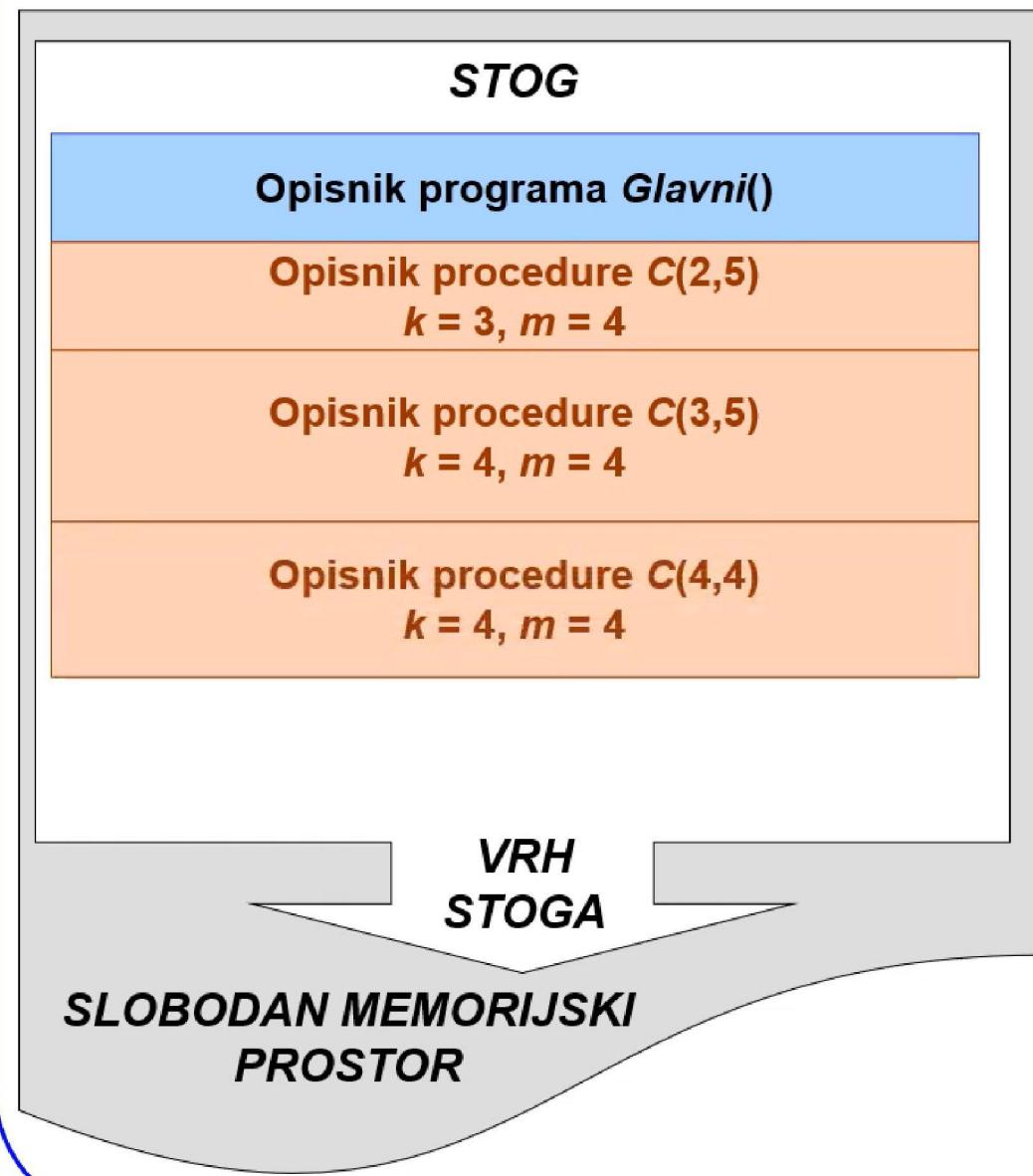
}

*C(2,5)*;

{

}





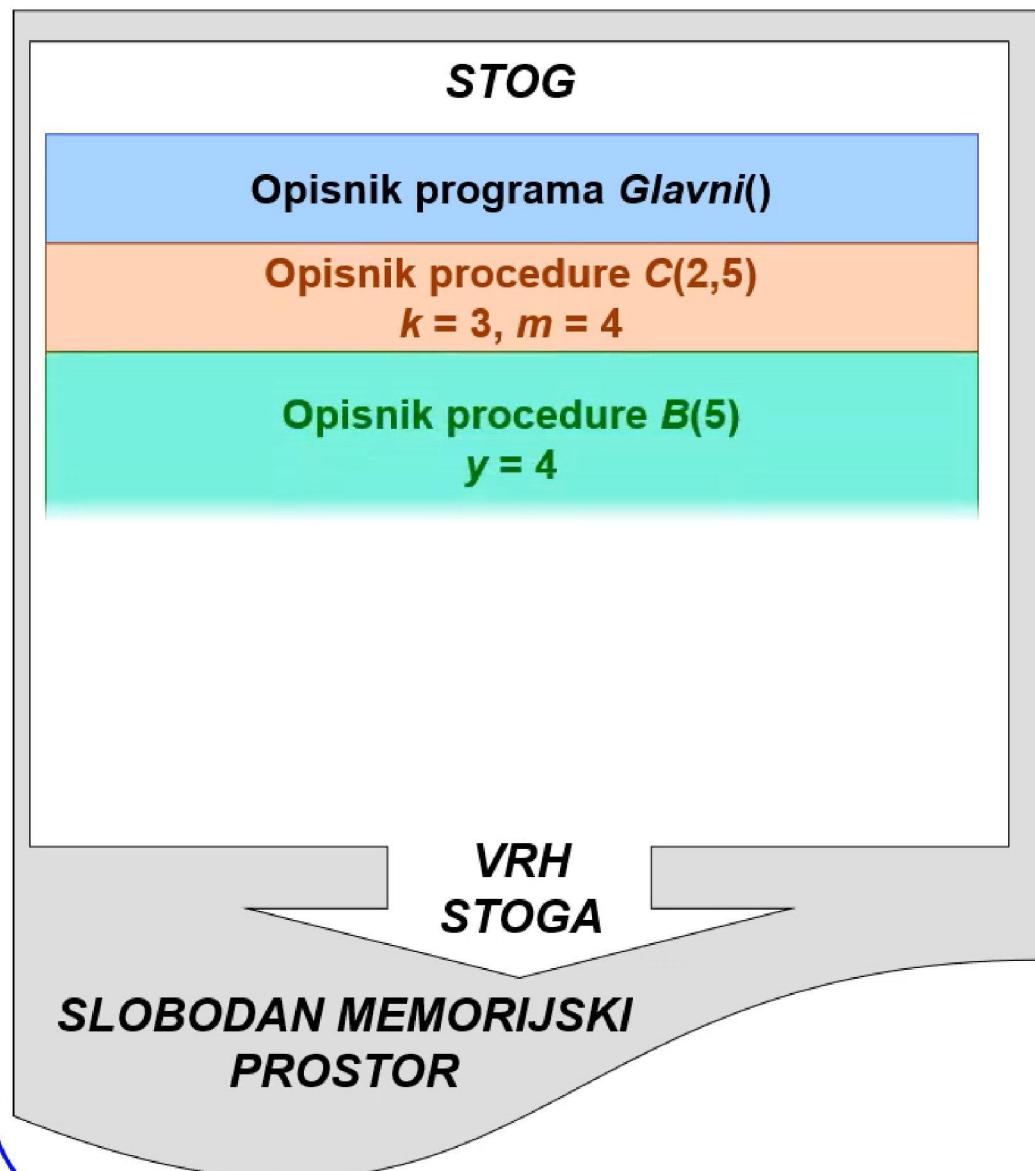
*Glavni()*

```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```



*Glavni()*

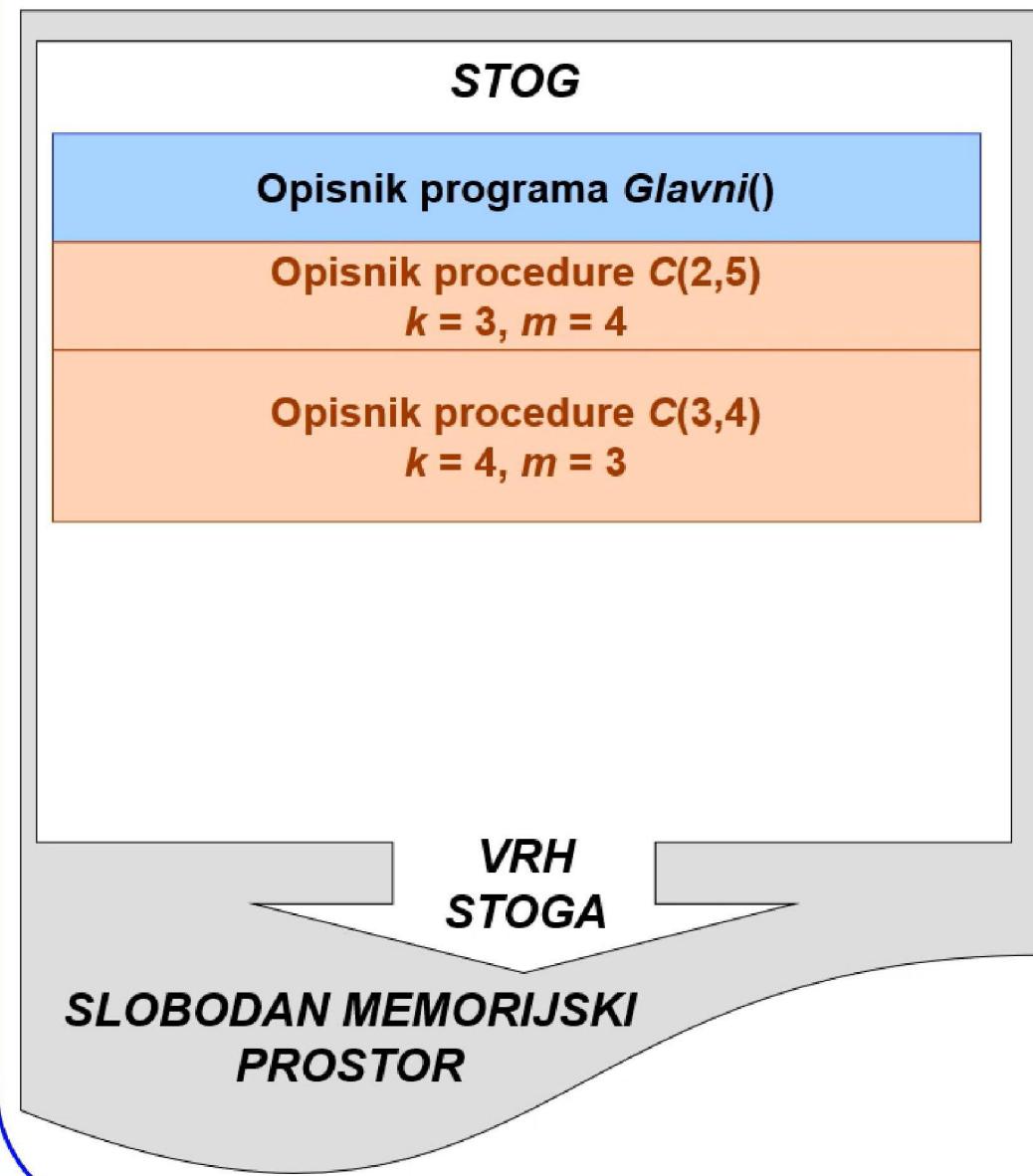
```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```





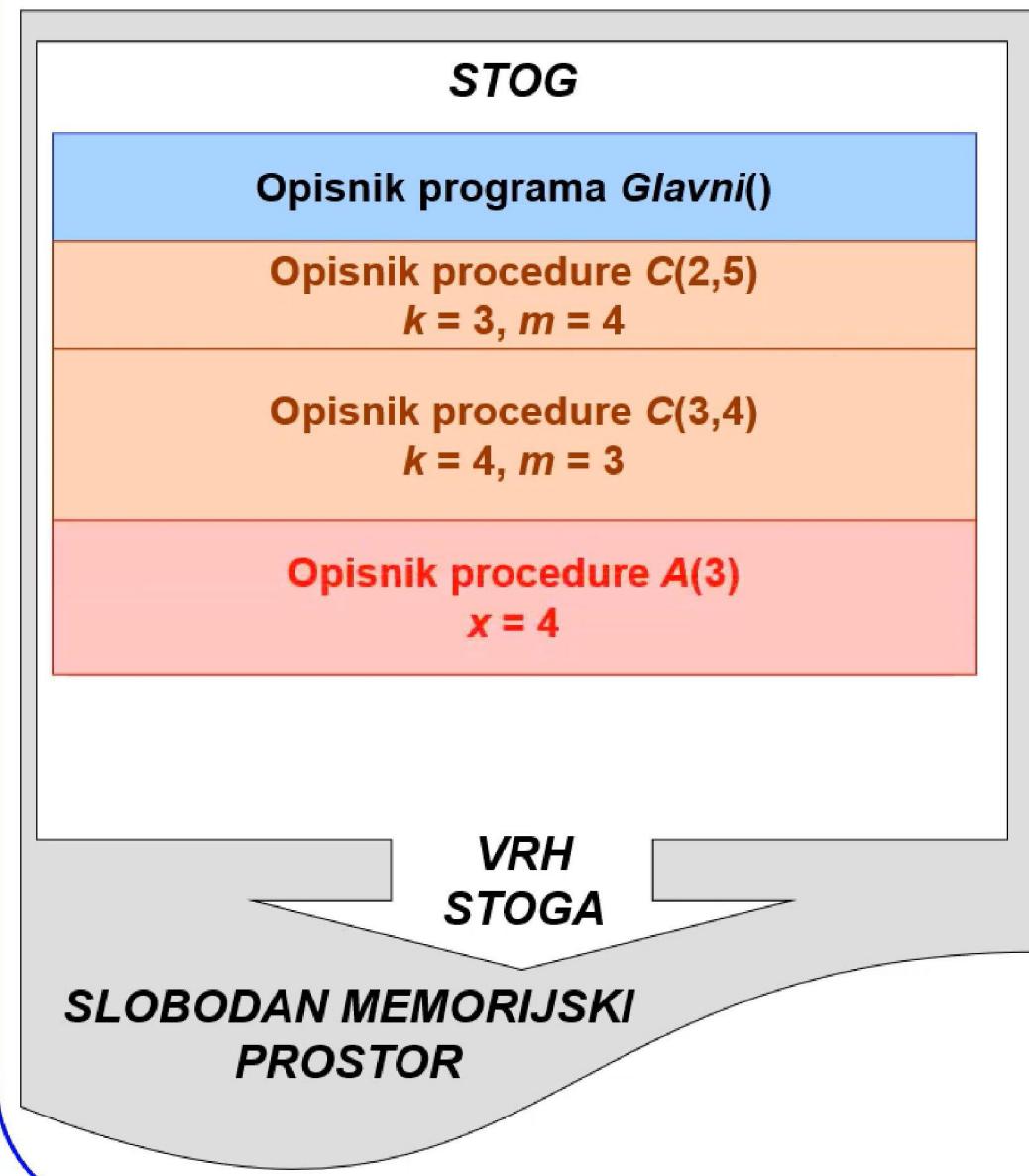
*Glavni()*

```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```



***Glavni()***

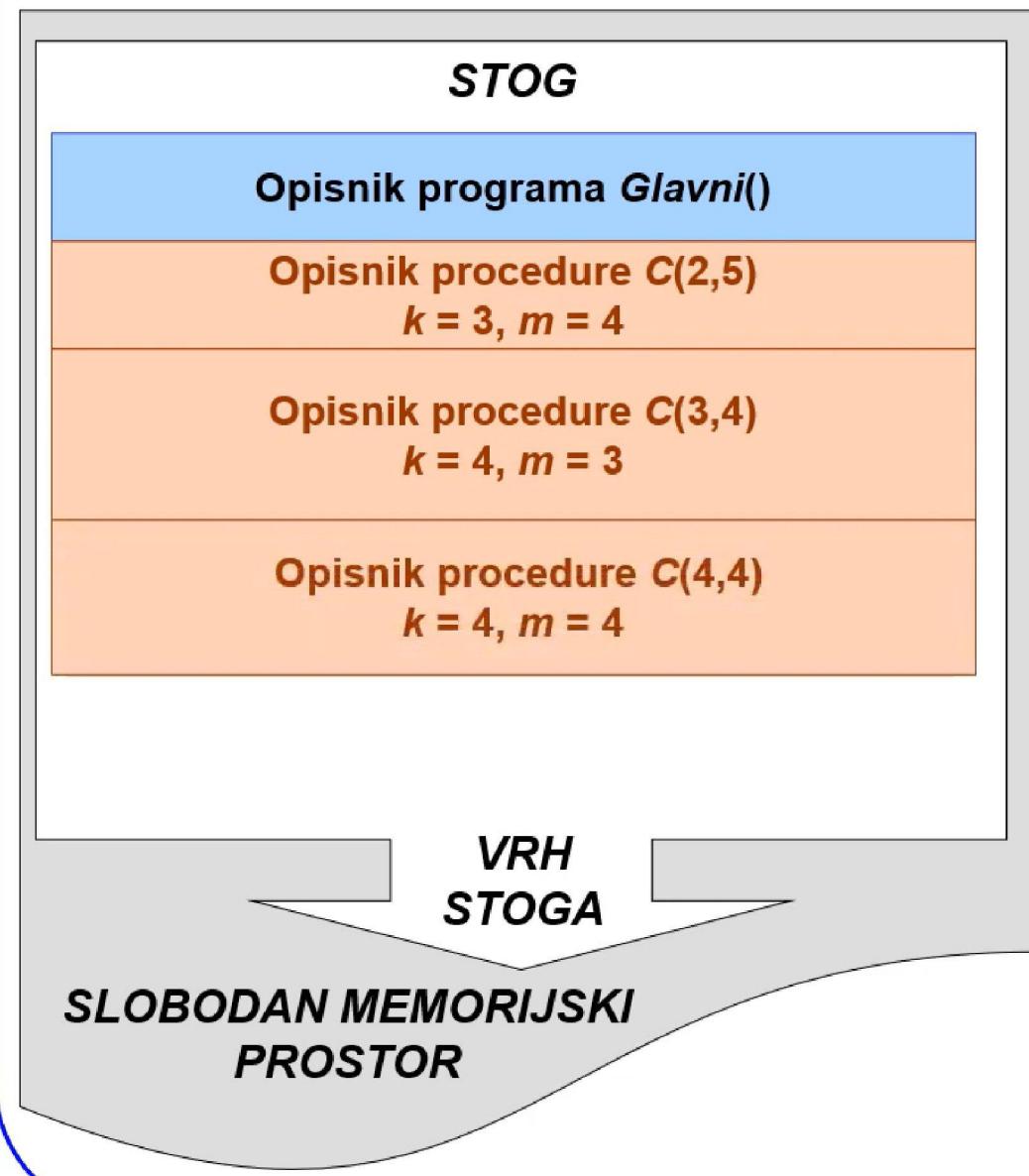
```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}

```



*Glavni()*

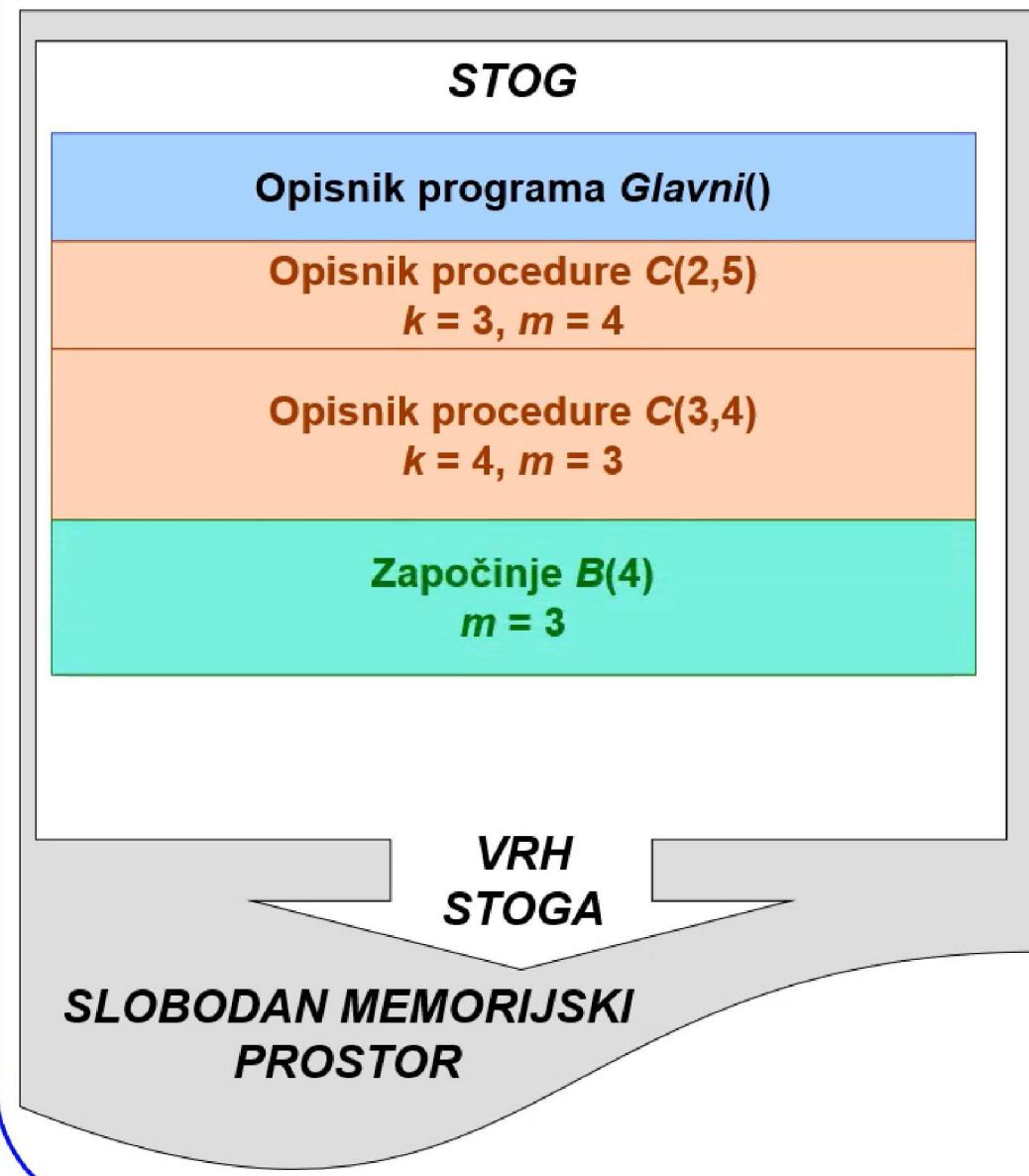
```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```





*Glavni()*

*A(x)*  
{

$x = x+1;$   
vrati (x);

}

*B(y)*  
{

$y = y-1;$   
vrati (y);

}

*C(k,m)*  
{

ako (k < m)  
{

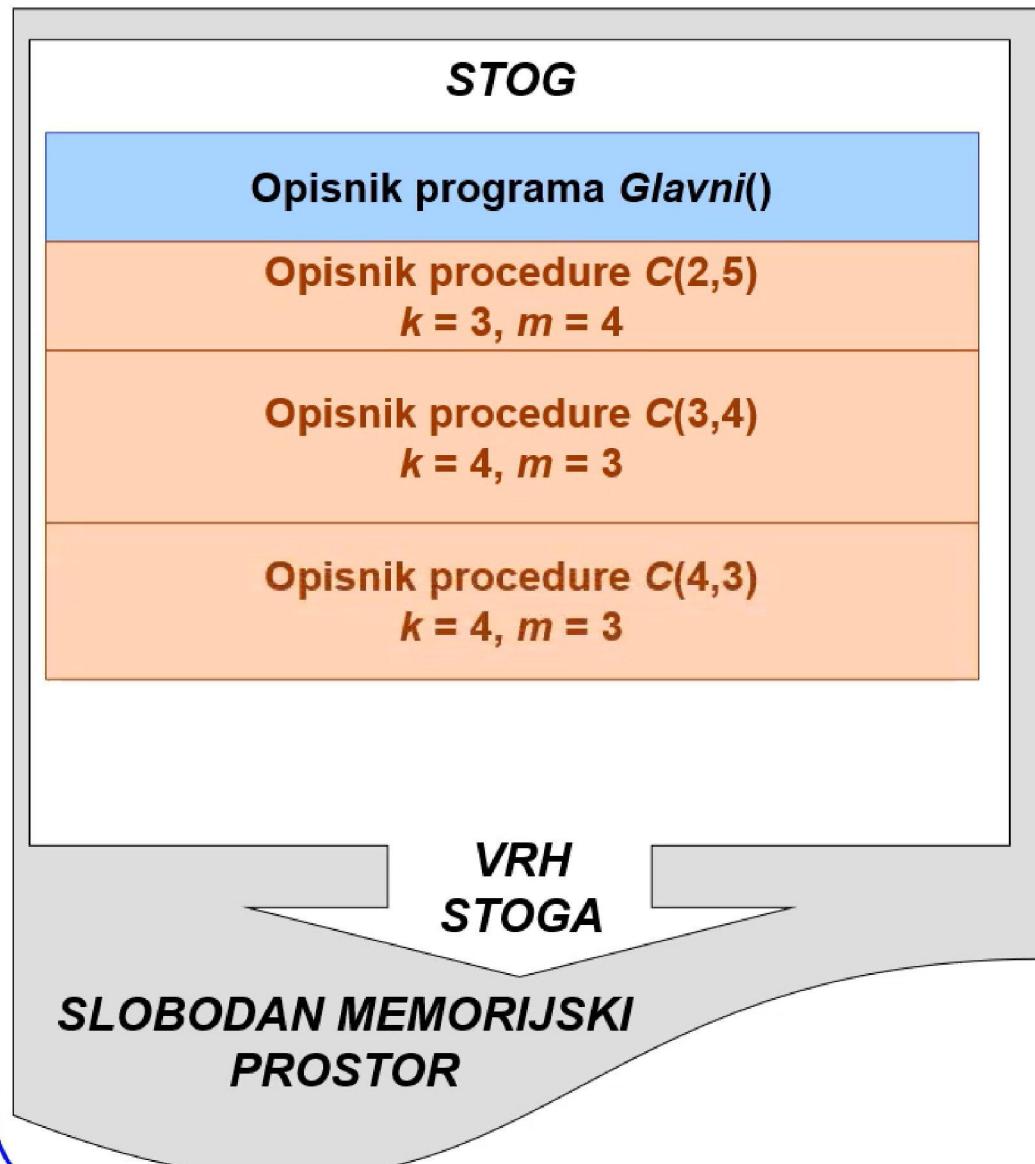
$k = A(k);$   
*C(k,m);*  
 $m = B(m);$   
*C(k,m);*

}

{  
}

*C(2,5);*





*Glavni()*

```

A(x)
{
  x = x+1;
  vrati (x);

}
B(y)
{
  y = y-1;
  vrati (y);

}
C(k,m)
{
  ako (k < m)
  {
    k = A(k);
    C(k,m);
    m = B(m);
    C(k,m);
  }
}
C(2,5);
}
```

# Dodjela memorije gomile

- **Gomila**
  - dio memorije
  - dinamički se dodjeljuje tijekom izvođenja programa
  - primjena stoga nije moguća:
    1. vrijednosti lokalnih podataka koriste se nakon završetka izvođenja procedure
    2. pozvana procedura ostaje aktivna nakon završetka izvođenja pozivajuće procedure



# GOMILA

Započinje **B(4)** -  $m = 3$

Opisnik procedure **C(4,4)** -  $k = 4, m = 4$

Opisnik procedure **A(3)** -  $x = 4$

Opisnik procedure **C(3,4)** -  $k = 4, m = 3$

Opisnik procedure **B(5)** -  $y = 4$

Opisnik procedure **C(4,4)** -  $k = 4, m = 4$

Opisnik procedure **B(5)** -  $y = 4$

Opisnik procedure **C(5,4)** -  $k = 5, m = 4$

Opisnik procedure **B(5)** -  $y = 4$

Opisnik procedure **C(5,5)** -  $k = 5, m = 5$

Opisnik procedure **A(4)** -  $x = 5$

Opisnik procedure **C(4,5)** -  $k = 5, m = 4$

Opisnik procedure **A(3)** -  $x = 4$

Opisnik procedure **C(3,5)** -  $k = 4, m = 4$

Opisnik procedure **A(2)** -  $x = 3$

Opisnik procedure **C(2,5)** -  $k = 3, m = 4$

Opisnik programa **Glavni()**

**Glavni()**

**A(x)**

{

$x = x+1;$   
vrati (x);

}

**B(y)**

{

$y = y-1;$   
vrati (y);

}

**C(k,m)**

{

ako ( $k < m$ )  
{

$k = A(k);$   
 $C(k,m);$   
 $m = B(m);$   
 $C(k,m);$

}

}

**C(2,5);**

