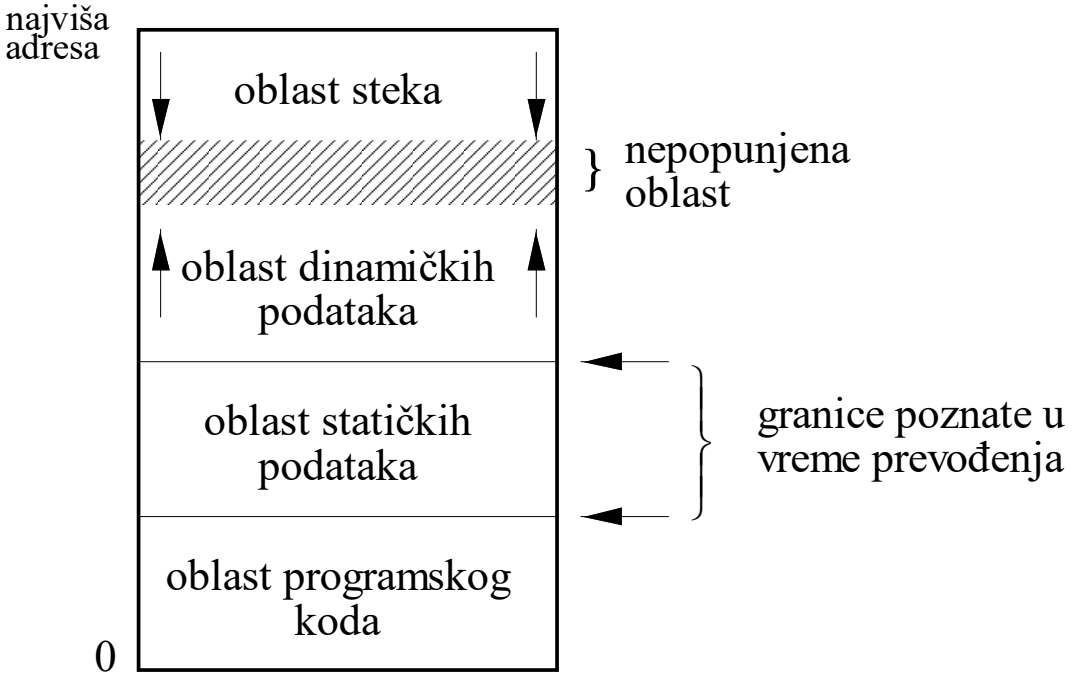
Izvršno okruženje

Uvod

- Definiše se kao podrška pri izvršavanju programa na višem programskom jeziku.
- Obuhvata sledeća razmatranja:
  - **Izvršna organizacija memorije** -- kako se predstavljaju različiti tipovi podataka i kako se dodeljuje memorija promenljivama različitih memorijskih klasa u programu? Kako se pristupa ovim promenljivama?
  - **Pravila vidljivosti** -- Kako upotreba određene promenljive uparuje sa odgovarajućom deklaracijom? Kako se iz određene procedure pristupa promenljivoj koja nije definisana u toj proceduri (nelokalnoj promenljivoj)?
  - **Poziv procedura** – Kako se generiše kod za poziv i povratak iz potprograma? Kako se pri pozivu procedura prenose stvarni parametri? Kako se realizuju rekurzivne procedure?
  - **Itd** (podrška obradi izuzetaka, podrška simboličkom debagovanju, podrška dinamičkom povezivanju...).

Izvršna organizacija memorije

- Jedna moguća podela operativne memorije u vreme izvršavanja programa:



- Za oblast steka najčešće se koristi procesorski stek (mada nije obavezno, na primer kada stek procesora nije u operativnoj memoriji).

- S obzirom da se veličine steka i oblasti dinamičkih podataka menjaju u toku izvršavanja programa, nije određena fiksna granica između ove dve oblasti već se svaka od njih može proširivati po potrebi do trenutka njihovog eventualnog međusobnog susreta.

- U ovom kontekstu termin **statički** generalno se koristi za veličine koje su poznate u vreme prevođenja programa (pre njegovog izvršavanja), dok se termin **dinamički** odnosi na veličine koje nisu poznate u vreme prevođenja, to jest, menjaju se tokom izvršavanja programa.

Memorijske klase promenljivih

- Vrednost neke promenljive u toku izvršavanja programa može se čuvati na različitim mestima:
  - u procesorskom registru (takozvane **registarske promenljive**);
  - u oblasti statičkih podataka (takozvane **statičke promenljive**);
  - u oblasti steka (takozvane **automatske promenljive**) ili
  - u oblasti dinamičkih podataka (takozvane **dinamičke promenljive**).
- Drugim rečima, promenljive poseduju različite memorijske klase (engl. *storage class*).

Memorijske klase promenljivih u nekim programskim jezicima

- **U Pascal-u**, na primer, promenljive deklarisane u glavnom programu smeštaju se statički, lokalne promenljive procedura (uključujući i parametre) smeštaju se najčešće na steku dok programer može eksplicitno alocirati (zauzeti) deo memorije iz dinamičke oblasti korišćenjem standardne funkcije new.
- **U C-u** se globalne promenljive (deklarisane izvan procedura) smeštaju statički, dok se lokalne promenljive procedura (i njihovi parametri) smeštaju na steku, a standardnom funkcijom malloc programer može eksplicitno alocirati deo memorije iz dinamičke oblasti. Programer može uticati na smeštanje promenljive navođenjem, u deklaraciji promenljive, takozvanog specifikatora memorijske klase (register, static ili auto).

Način pristupa skalarnoj (na primer, celobrojnoj) promenljivoj

- zavisi od njene memorijske klase:
  - **Registarskim promenljivama** se pristupa **registarskim direktnim adresiranjem**.
  - **Statičkim promenljivama** se pristupa **memorijskim direktnim adresiranjem** s obzirom da su veličina i početna adresa smeštanja promenljive u memoriji poznati u vreme prevođenja programa.
  - **Automatskim promenljivama** se pristupa **baznim adresiranjem**, što će detaljnije biti objašnjeno u nastavku.
  - **Dinamičkim promenljivama** se pristupa nekom vrstom **indirektnog adresiranja**. Adrese ovih promenljivih ne mogu biti ugrađene u programski kod s obzirom da nisu poznate u vreme prevođenja.

Studija slučaja: Arhitektura procesora Intel 80x86

- 8086 poseduje 16-bitne registre opšte namene AX, BX, CX i DX i 16-bitne indeksne registre SI i DI.
- Adresiranje je bajtovsko, a na steku se čuvaju isključivo 16-bitne veličine.
- Kod mikroprocesora Intel 8086 za rad sa stekom koriste se 16-bitni registri SP (*Stack Pointer*) i BP (*Base Pointer*).
- Stek raste prema nižim memorijskim lokacijama i registar SP ukazuje na poslednju popunjenu lokaciju.

8086- Izbor iz seta instrukcija

Instrukcija	Objašnjenje
PUSH r/m	stavlja vrednost operanda na stek
POP r/m	skida vrednost sa vrha steka i dodeljuje je operandu
CALL r/m	Stavlja tekuću vrednost brojača naredbi IP na stek i vrši skok na adresu specificiranu operandom
RET [c]	SP uvećava za vrednost specificiranu operandom c; skida vrednost sa vrha steka i dodeljuje brojaču naredbi IP
ADD dst, src	Sabira vrednost operanda src sa operandom dst i rezultat smešta u dst
SUB dst, src	Oduzima vrednost operanda src sa operandom dst i rezultat smešta u dst
MOV dst, src	Dodeljuje operandu dst vrednost operanda src

Načini adresiranja kod 8086

Primeri sintakse	Objašnjenje	Naziv
100	vrednost operanda je 100	Neposredno
labela [100]	adresa operanda je zadata vrednost	memorijsko direktno
AX BP	vrednost operanda je sadržaj registra	registarsko direktno
[BP] [SI] [DI]	adresa operanda je sadržaj nekog od registara BP, SI, DI.	registarsko indirektno
[BP+4] [DI-100] [SI-20]	adresa operanda je sadržaj registra ± konstanta	bazno ili indeksno

Primer

- U nekoj C proceduri promenljive r, s, a i d deklarisane su na sledeći način:

```
register int r;   /* registarska celobrojna promenljiva */
static int s;     /* statička celobrojna promenljiva */
auto int a;       /* automatska celobrojna promenljiva */
static int *d;    /* statička promenjiva tipa pokazivača na ceo broj. */
```

- Pod pretpostavkom da je dužina celobrojnog podatka 16-bita i da se pokazivač d inicijalizuje pozivom `d=(int *)malloc(sizeof(int))`, koji asemblerski kod odgovara naredbama dodele:

```
r=s;
a=*d;  ?
```

Rešenje

r = s

generiše sledeći asemblerski kod:

```
MOV    SI,[0200]
```

Izvorišni (desni) operand zadat je memorijskim direktnim, a odredišni registarskim adresiranjem.

a=*d;

Ovoj naredbi odgovara sledeći asemblerski kod:

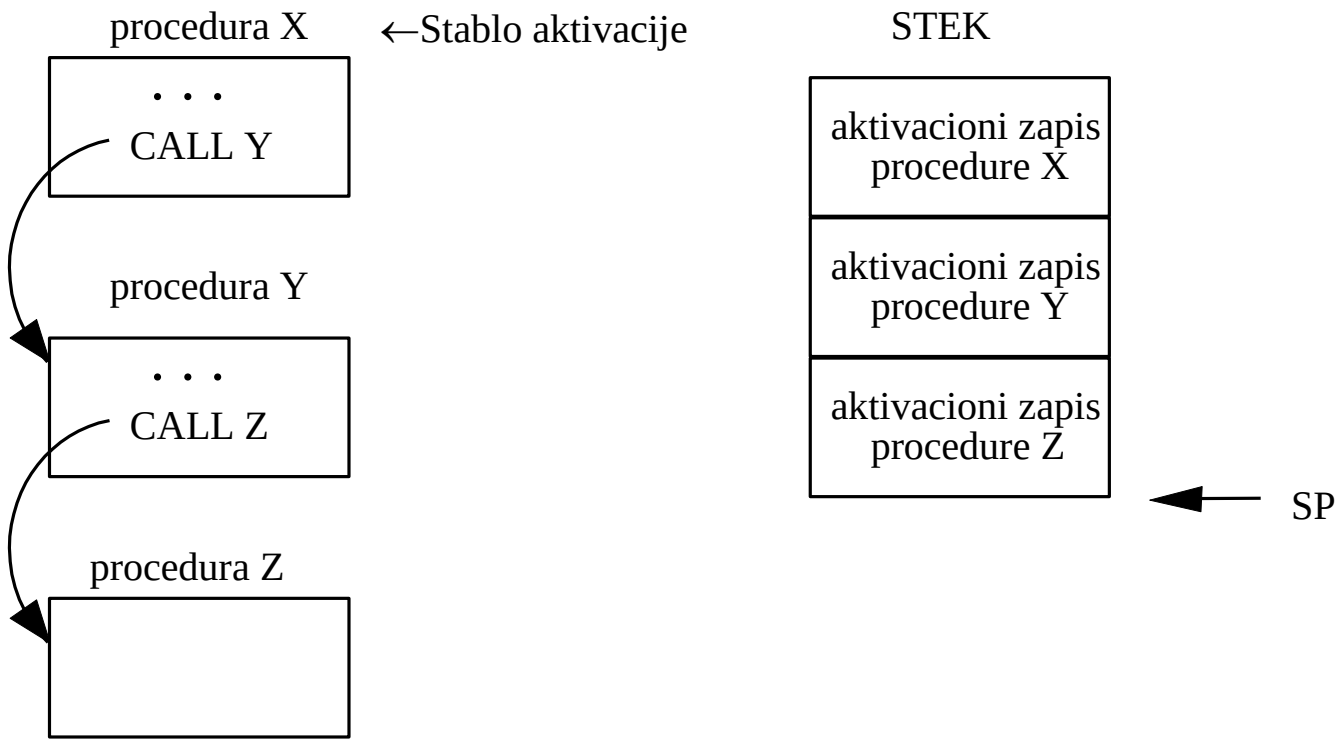
```
MOV BX,[0202]          ; u BX ide sadržaj promenljive d
MOV AX,[BX]            ; u AX ide sadržaj *d (sa heapa)
MOV [BP-02],AX         ; a:=*d
```

Korišćenje registra AX pri ovom transferu je bilo neophodno, jer ne postoji instrukcija transfera iz memorije u memoriju (na primer `MOV [BP-02], [BX]`).

Pozivanje i povratak iz potprograma

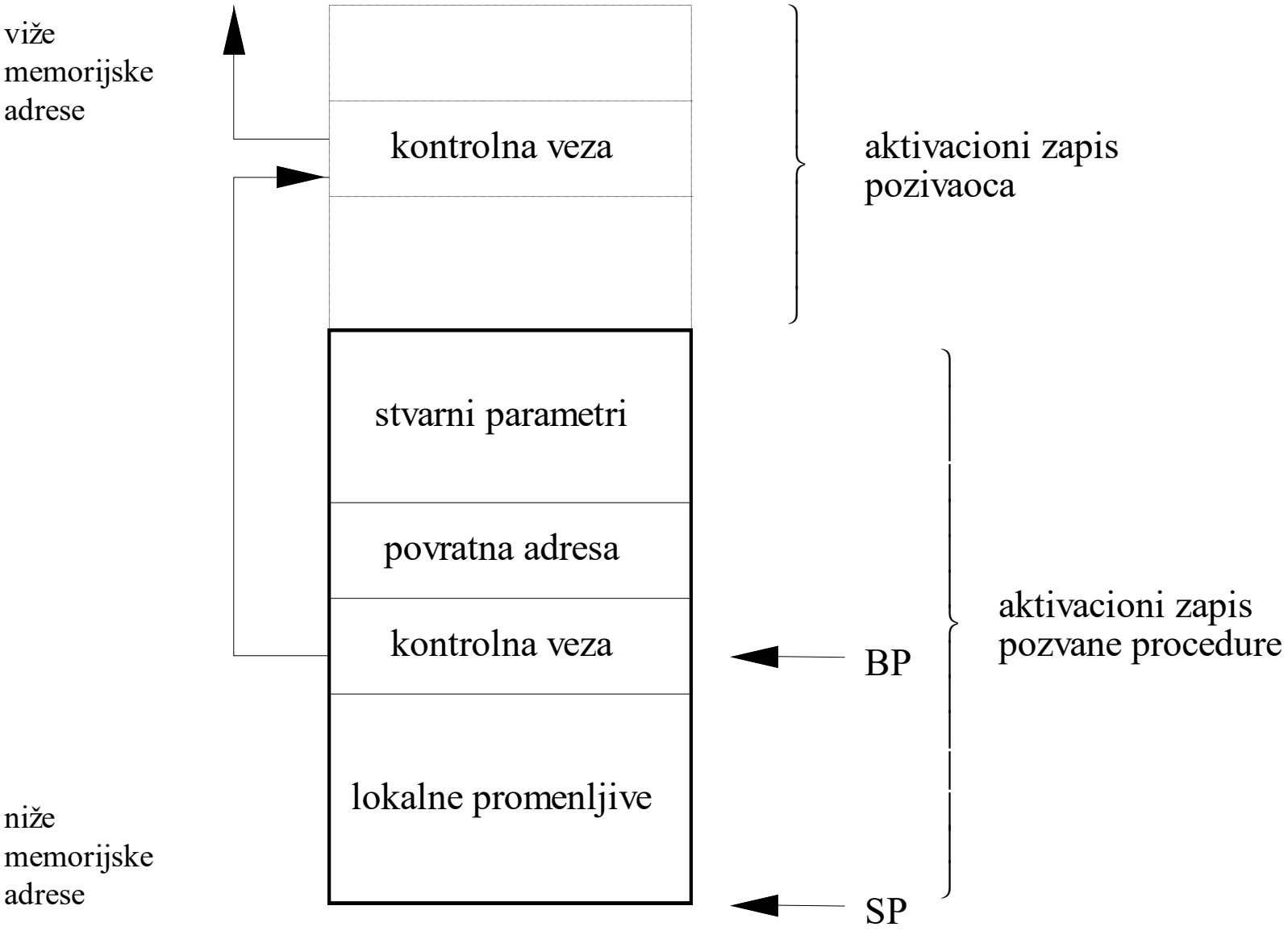
- Pri svakom pozivu procedure u toku izvršavanja programa, na vrh steka se stavlja blok podataka koje koristi pozvana procedura u svom radu. Ovaj blok zove se **aktivacioni zapis** procedure ili **okvir** (engl. *frame*).
- AZ na steku ostaje sve dok pozvana procedura ne vrati kontrolu pozivaocu, kada se AZ pozvane procedure skida sa vrha steka.

Primer



Struktura aktivacionog zapisa

- za C procedure



Sekvenca pozivanja

- Niz aktivnosti koje pri pozivu neke procedure izvršavaju pozivalac i pozvana procedura da bi uspostavili aktivacioni zapis pozvane procedure naziva se **sekvenca pozivanja**.

- **Procedura-pozivalac** započinje:
- Na stek se redom stavljaju stvarni parametri za pozvanu proceduru.
- Izvršava se CALL naredba koja na stek stavlja povratnu adresu i prenosi kontrolu pozvanoj proceduri

- **Pozvana procedura**:
- Na stek stavlja sadržaj registra BP (radi se o polju kontrolne veze).
- Sadržaj registra SP, koji u tom trenutku ukazuje na kontrolnu vezu, kopira u BP.
- Od sadržaja registra SP oduzima broj N koji je poznat u vreme prevođenja. Radi se o broju memorijskih lokacija potrebnih za smeštaj lokalnih promenljivih. Na ovaj način se na steku alocira potreban prostor za lokalne promenljive.
- Nakon završetka sekvence pozivanja, izvršava se kod pozvane procedure za inicijalizaciju lokalnih promenljivih itd.

Sekvenca povratka

- Prilikom povratka iz pozvane u pozivajuću proceduru izvršava se **sekvenca povratka**.

- **Pozvana procedura**:

- Sadržaj registra BP kopira se u registar SP čime se sa steka oslobađa prostor za lokalne promenljive i one efektivno prestaju da postoje.
- Sa vrha steka (koji u tom trenutku ukazuje na polje kontrolne veze pozvane procedure) skida se vrednost kontrolne veze i smešta u registar BP. Time je ovaj registar postao spreman za upotrebu u proceduri-pozivaocu.
- Naredba RET skida sa vrha steka povratnu adresu i smešta u brojač naredbi IP. Time se kontrola predaje pozivaocu.

- **Procedura - pozivalac**:

- Dodavanjem odgovarajuće vrednosti na SP sa steka uklanja stvarne parametre i normalno nastavlja rad.

Primer

Napisati 80x86 asemblerski potprogram ekvivalentan datom programu na C-u. Koja sekvenca asemblerskog koda odgovara pozivu `primer(10,20)` u glavnom programu?

```
void primer(int a,b) {
  int c;
  c=a;
}
```

Rešenje

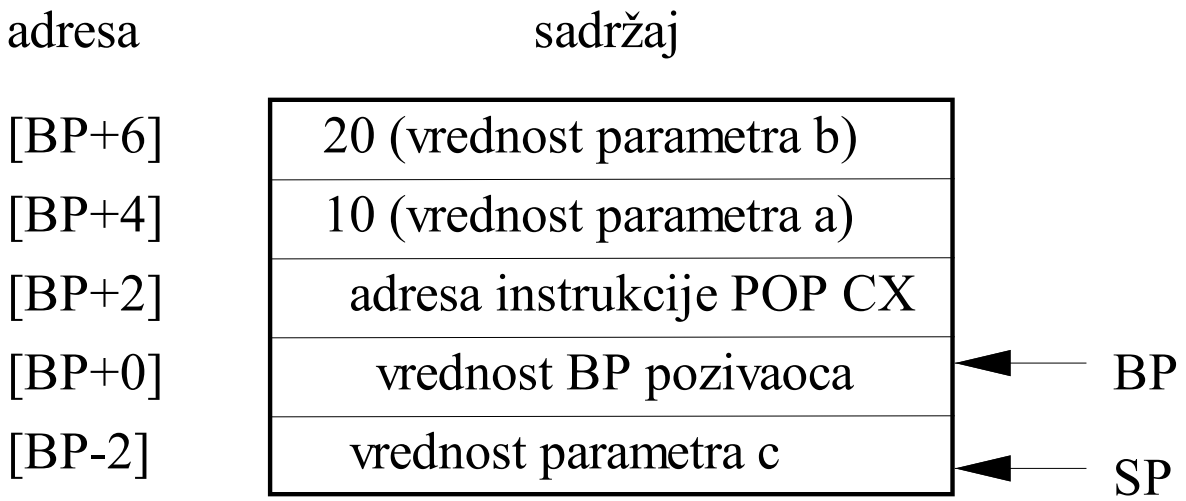
Poziv primer(10,20) prevodi se sa:

```
MOV AX,20
PUSH    AX      ; smeštanje drugog parametra na stek
MOV AX,10
PUSH    AX      ; smeštanje prvog parametra na stek
CALL    primer  ; poziv procedure primer
POP CX          ; skidanje parametara sa steka
POP CX          ; alternativa navedenim je operacija ADD SP,4
```

Kod za proceduru primer je:

```
PUSH    BP      ; smeštanje sadržaja BP u polje kontrolne veze
MOV BP,SP       ; nova vrednost BP treba da ukazuje na kontrolnu vezu
SUB SP,2        ; alociranje prostora za lokalnu promenljivu c
MOV AX,[BP+04]  ; u AX ide vrednost stvarnog parametra a
MOV [BP-02],AX  ; u lokalnu promenljivu c ide vrednost iz AX
MOV SP,BP       ; oslobađanje prostora dodeljenog promenljivoj c
POP BP          ; restauracija vrednosti BP iz kontrolne veze
RET             ; povratak u pozivajuću proceduru
```

Na slici je prikazan izgled **aktivacionog zapisa procedure primer**.

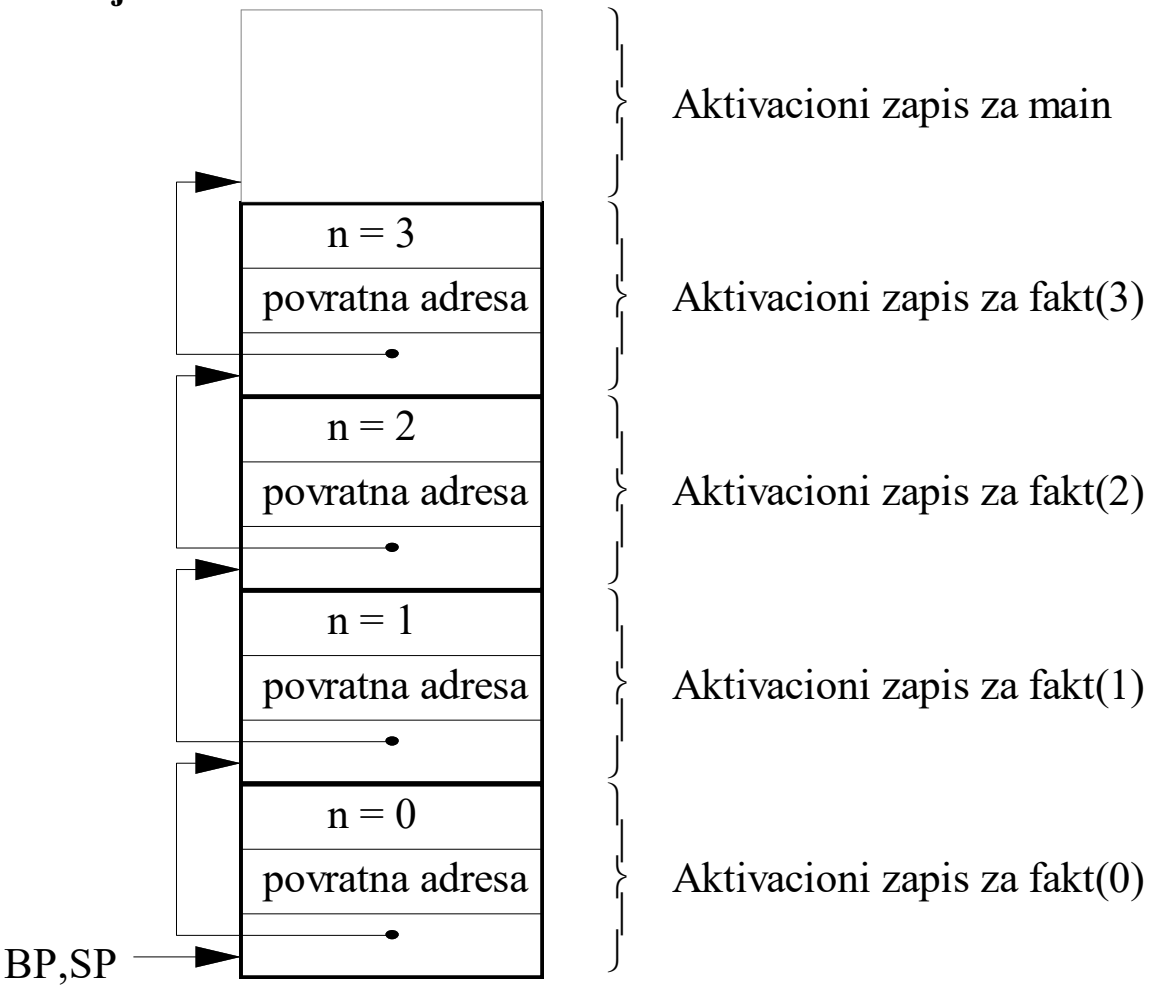


Rekurzija: primer

Data je C funkcija `proba`. Pratiti tok izvršavanja poziva `fakt(3)` i odrediti izgled procesorskog steka u trenutku neposredno pre prvog povratka iz funkcije.

```
int fakt(int n) {
    return (n>0)?(n * fakt(n-1)):1;
}
```

Rešenje

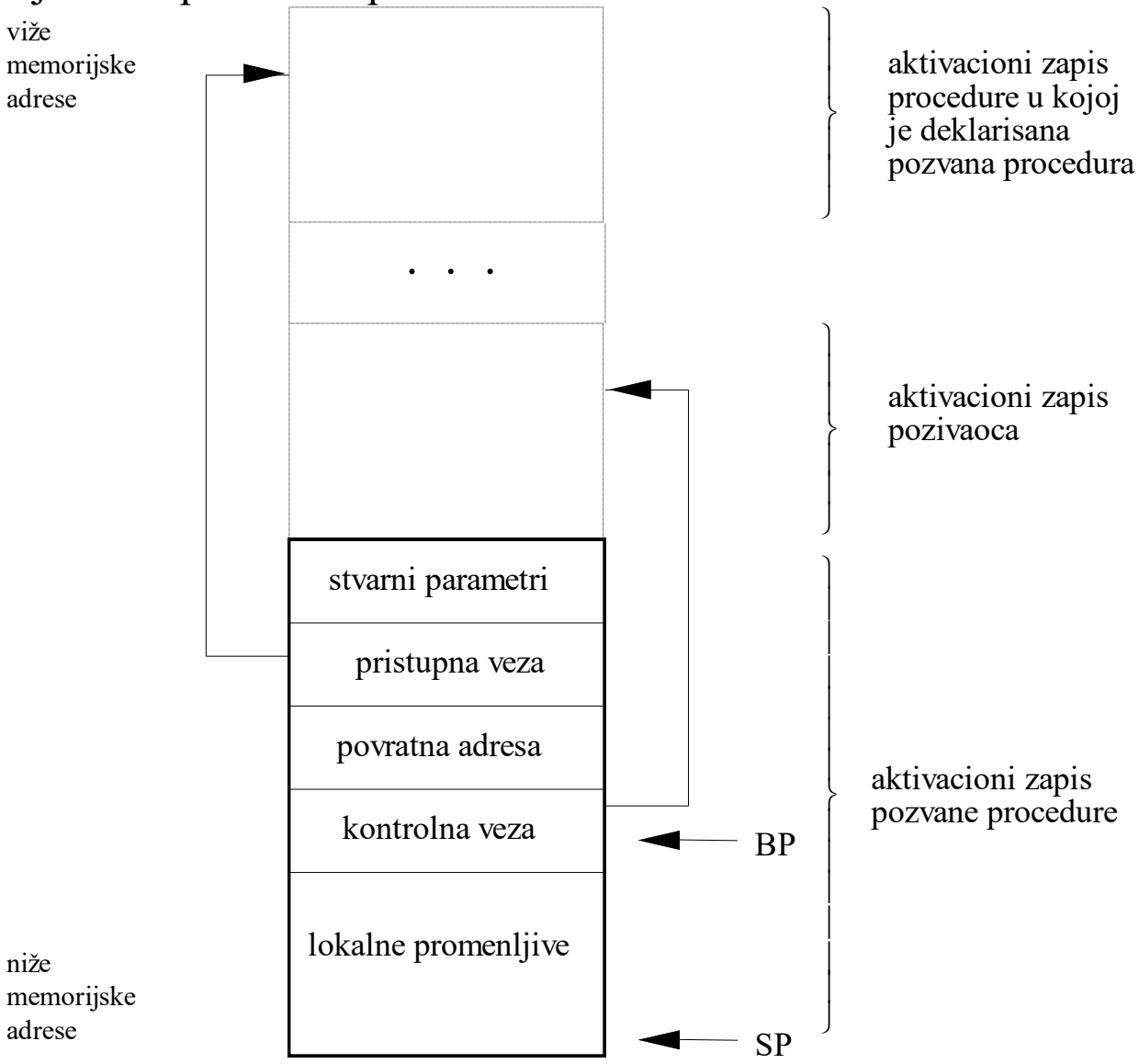


Statičko okruženje za nelokalne promenljive realizovano pristupnim vezama

- U Pascal-u je, za razliku od C-a, dozvoljeno ugneždavanje deklaracija procedura.
- Prema pravilu leksičkog opsega važenja deklaracije promenljivih, u nekoj Pascal-skoj proceduri, pored lokalnih promenljivih, vidljive su i sve promenljive deklarisane u okružujućim procedurama ako nemaju isto ime kao lokalne promenljive.
- U odnosu na aktivacioni zapis za C procedure, razlika je u polju **pristupne veze**. Polje pristupne veze potrebno je da bi se realizovao pristup nelokalnim promenljivama koje su deklarisane u drugim procedurama.

Aktivacioni zapis sa pristupnom vezom

- Polje **pristupne veze** neke procedure ukazuje na AZ (preciznije, na polje kontrolne veze toga zapisa) neposredno okružujuće procedure (u listingu programa). Ova procedura, u opštem slučaju, razlikuje se od procedure pozivaoca.



Algoritam formiranja pristupne veze

- Razmotrimo tri moguća slučaja kada Pascal-ska procedura A poziva proceduru B:

i) Deklaracija B je ugneždena u deklaraciji A; kaže se da je B deklarisana na (za jedan) višem *leksičkom nivou ugneždavanja* od A.

```
procedure A;
        procedure B; begin...end;
begin B end.
```

U ovom slučaju, pristupna veza procedure B ukazivaće na AZ(A).

ii) Deklaracija B je ugneždena u istoj proceduri X (ili u glavnom programu) kao i deklaracija A; tada važi da su A i B na istom leksičkom nivou ugneždavanja.

```
procedure X;
                    procedure B; begin...end;
                    procedure A; begin B end;
begin ... end.
```

U ovom slučaju, pristupna veza A kopira se u polje pristupne veze B, tako da obe ukazuju na AZ(X).

iii) Leksički nivo ugneždavanja A je veći od nivoa B. Razlika leksičkih nivoa A i B označena sa N može da bude veća od jedan;

```
procedure X
  procedure B;
     procedure Y
                    procedure A; begin B end;
    begin A; end;
  begin Y; end;
begin B; end;
```

u ovom slučaju potrebno je slediti lanac od pristupnih veza počev od pristupne veze A i u polje pristupne veze za B iskopirati sadržaj N-tog polja u lancu. (Za N=1 to je polje na koje ukazuje polje pristupne veze za A, itd.)

Pristupne veze - sekvenca pozivanja

- Različite obaveze u odnosu na C procedure:
- **Procedura pozivalac**:
- na stek redom stavlja stvarne parametre (uobičajeno za Pascal je da kranje levi parametar ide prvi)
- na stek stavlja vrednost koja definiše sadržaj polja pristupne veze pozvane procedure prema gore navedenim pravilima (na primer, u slučaju i, na stek se stavlja sadržaj registra BP)
- Izvršava se CALL naredba koja na stek stavlja povratnu adresu i prenosi kontrolu pozvanoj proceduri
- **Pozvana procedura**:
- Na stek stavlja sadržaj registra BP (radi se o polju kontrolne veze).
- Sadržaj registra SP, koji u tom trenutku ukazuje na kontrolnu vezu, kopira u BP.
- Od sadržaja registra SP oduzima broj N koji je poznat u vreme prevođenja. Radi se o broju memorijskih lokacija potrebnih za smeštaj lokalnih promenljivih. Na ovaj način se na steku alocira potreban prostor za lokalne promenljive.
- Nakon završetka sekvence pozivanja, izvršava se kod pozvane procedure.

Pristupne veze - sekvenca povratka

- Prilikom povratka iz pozvane u pozivajuću proceduru izvršava se *sekvenca povratka*, koju **u potpunosti realizuje pozvana procedura**:
  - Sadržaj registra BP kopira se u registar SP čime se sa steka oslobađa prostor za lokalne promenljive i one efektivno prestaju da postoje.
  - Sa vrha steka (koji u tom trenutku ukazuje na polje kontrolne veze pozvane procedure) skida se vrednost kontrolne veze i smešta u registar BP. Time je ovaj registar postao spreman za upotrebu u proceduri-pozivaocu.
  - Naredba RET *n* skida sa vrha steka povratnu adresu i smešta u brojač naredbi IP. Ova naredba zatim uvećava registar SP za vrednost *n* čime se sa steka skidaju stvarni parametri i pristupna veza. Nakon toga se kontrola predaje pozivaocu.

Pristupne veze - primer

Odrediti 80x86 asemblerski ekvivalent datog Pascal programa. Kako izgleda procesorski stek u trenutku neposredno pre prvog povratka iz procedure?

```
program glavni;
    procedure jedan
    var local1: integer;
        procedure tri; forward;
        procedure dva;
        begin {dva}
            tri  {II slučaj kod formiranja pristupne veze}
        end; {dva}
        procedure tri;
        begin {tri}
            local1:=10;
        end; {tri}
    begin {jedan}
        dva {I slučaj kod formiranja pristupne veze}
    end; {jedan}
begin {glavni}
    jedan

end. {glavni}
```

Rešenje

procedura dva

```
; sadržaj registra BP definise sadržaj polja kopntrolne veze
PUSHBP
; BP ukazuje na polje kontrolne veze
MOV BP,SP
; počinje sekvenca poziva procedure tri
; uspostavlja se sadržaj polja kontrolne veze u aktivacionom zapisu
; procedure tri,
; kopiranjem sadržaja polja kontrolne veze u akt. zapisu procedure dva
; (ii slučaj algoritma)
PUSH[BP+04]
; poziv procedure tri
CALL tri
; sekvenca povratka iz procedure dva
; dealocira se deo aktivacionog zapisa iza polja kontrolne veze
MOV SP,BP
; restaurira se sadržaj registra BP potreban pozivaocu i dealocira
; polje kontrolne veze sa steka
POP BP
; povratak pozivaocu uz dealociranje polja pristupne veze
RET 0002
```

procedura tri

```
; definisanje polja kontrolne veze
PUSHBP
; BP ukazuje na polje kontrolne veze
MOV BP,SP
; prevod naredbe local1 := 10
MOV DI,[BP+04]
MOV [DI-02],10*
; sekvenca povratka
MOV SP,BP
POP BP
RET 0002
```

---

* Na 8086 asembleru sintaksa ove instrukcije glasi MOV SS:WORD PTR [DI-2],10
SS: označava da se pristupa segmentu steka (podrazumevana vrednost za registar DI je segment podataka DS),
WORD PTR označava da se prenosti jedna reč, a ne jedan bajt.

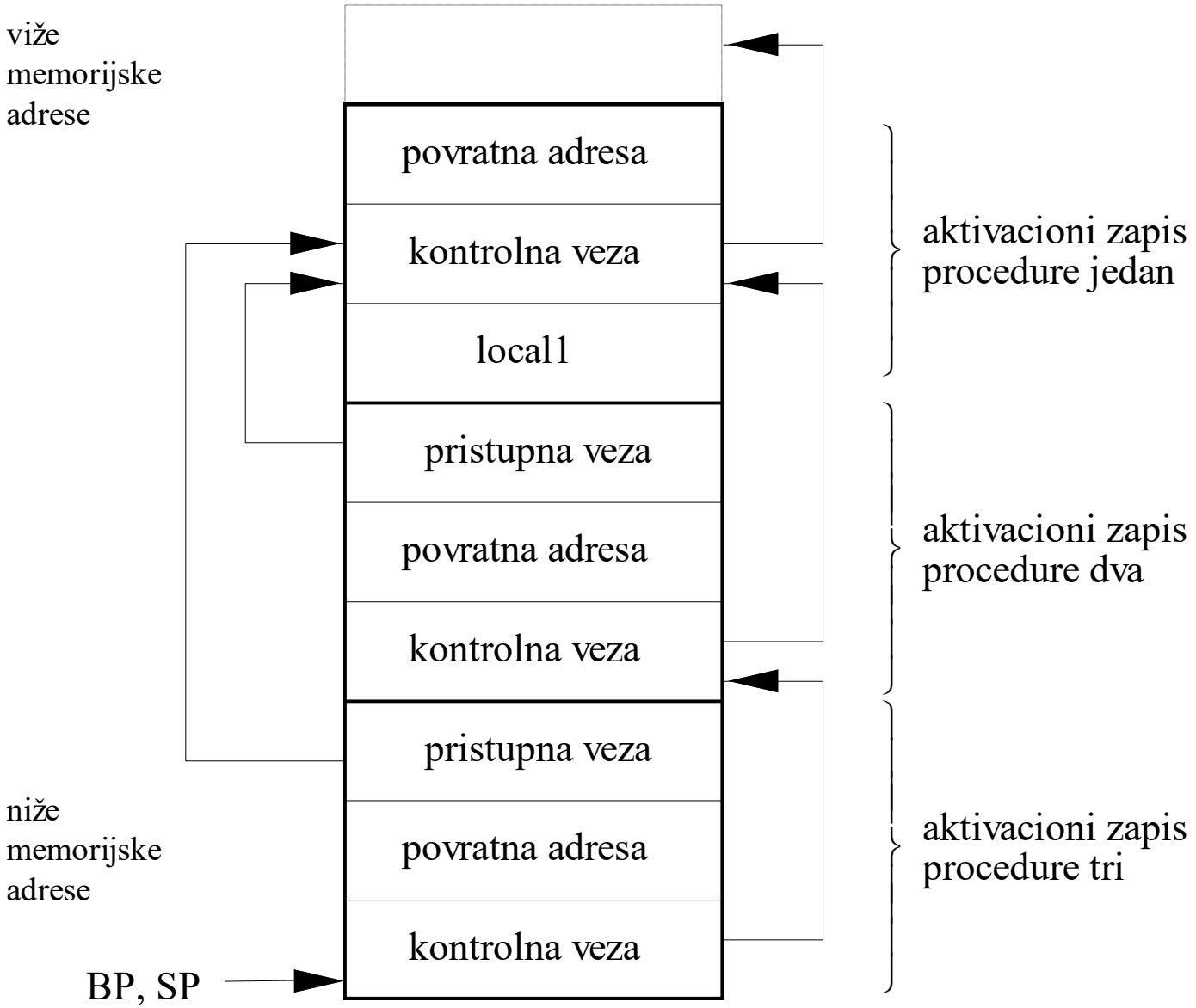
procedura jedan

```
; definisanje polja kontrolne veze
PUSH    BP
; BP ukazuje na polje kontrolne veze
MOV BP,SP
; alociranje prostora za promenljivu local1
SUB SP,+02
; definisanja pristupne veze za proceduru dva (slučaj i)
PUSH    BP
; poziv procedure dva
CALL    dva
; sekvenca povratka
MOV SP,BP
POP BP
RET
```

glavni program - sekvenca poziva

```
; polje pristupne veze nije potrebno proceduri jedan pošto su
; promenljive glavnog programa statički alocirane
CALL    jedan
```

U vreme izvršavanja procedure tri, izgled procesorskog steka je prikazan na slici


više
memorijske
adrese
povratna adresa
kontrolna veza
local1
pristupna veza
povratna adresa
kontrolna veza
pristupna veza
povratna adresa
kontrolna veza
niže
memorijske
adrese
BP, SP
aktivacioni zapis
procedure jedan
aktivacioni zapis
procedure dva
aktivacioni zapis
procedure tri

Statičko okruženje realizovano displejima

- U opštem slučaju, ako se promenljivoj deklarisanoj na leksičkom nivou M pristupa iz procedure deklarisane na nivou N kao nelokalnoj promenljivoj (te je N > M), potrebno je slediti lanac od N-M pristupnih veza da bi se došlo do aktivacionog zapisa u kome se nalazi promenljiva.
- Efikasniji pristup nelokalnim promenljivama realizuje se korišćenjem **displeja**. Za svaki leksički nivo na kome se može nalaziti nelokalna promenljiva čuva se poseban pokazivač na aktivacioni zapis odgovarajuće procedure deklarisane na tom nivou; niz tih pokazivača naziva se **displej**.
- Za pristup nelokalnoj promenljivoj putem displeja uvek su dovoljne dve naredbe, bez obzira na kojem se leksičkom nivou nalazi neloklana promenljiva:
  - Pva naredba u registar dovodi sadržaj odgovarajueg pokazivača koji predstavlja adresu aktivacionog zapisa u kome se nalazi nelokalna promenljiva.
  - Druga naredba vrši pristup koristeći bazno indeksiranje sa poznatim pomerajem.
- Pokazivači displeja mogu se čuvati u registrima procesora, u statičkoj zoni memorije ili u aktivacionom zapisu procedure.

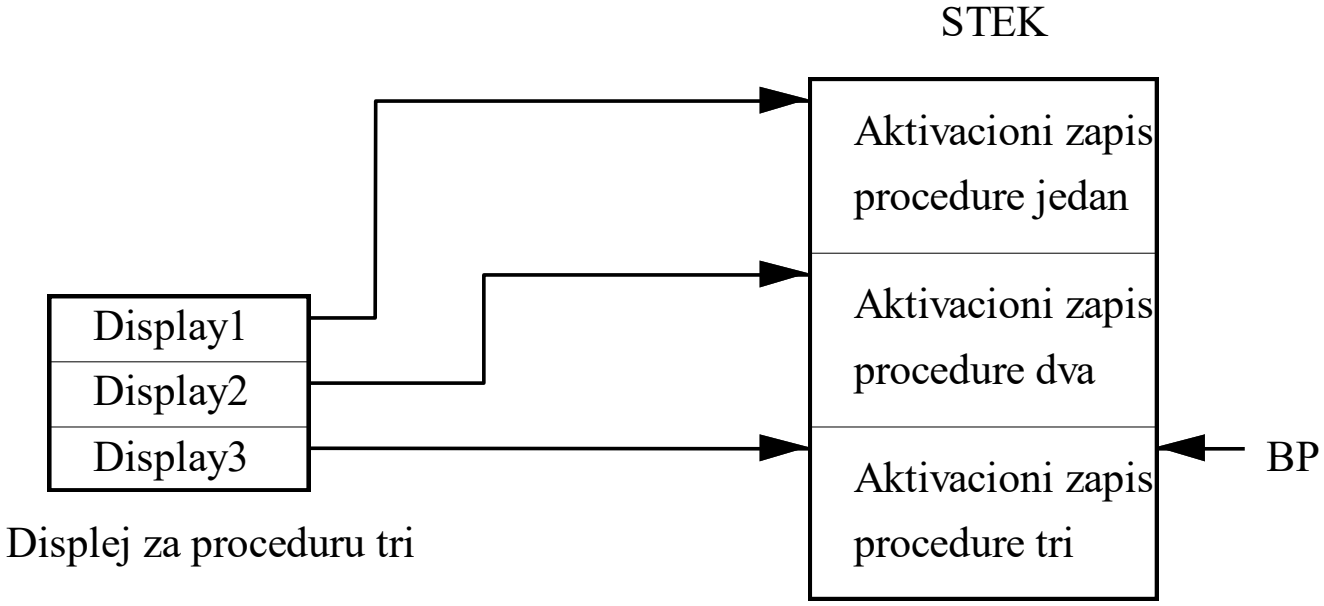
Displeji – Primer

Odrediti 80x86 asemblerski ekvivalent datog Pascal programa. Kako izgledaju displeji u trenutku neposredno pre prvog povratka iz procedure?

```
program glavni;
    procedure jedan;
    var local1: integer;
        procedure dva;
            procedure tri;
            begin {tri}
                local1:=10
            end; {tri}
        begin {dva}
            tri
        end; {dva}
    begin {jedan}
        dva
    end; {jedan}
begin {glavni}
    jedan
end. {glavni}
```

Rešenje

- procedura tri koja se nalazi na leksičkom nivou tri (glavni program je na nivou nula), imala bi tri[1] pokazivača u displeju kako je prikazano na slici **Error! Bookmark not defined.**.



- Pod pretpostavkom da su pokazivači statički alocirani, naredbi local1:=10 bi odgovarao sledeći segment asemblerskog koda:

```
MOV DI, Display1 ; u SI ide adresa aktivacionog zapisa procedure jedan
MOV [DI-4], 10   ; u promenljivu local1 se upisuje 10
                 ; (pomeraj se zna u vreme prevođenja)
```

- U slučaju pristupnih veza:

```
MOV DI,[BP+4] ; u DI ide sadržaj polja pristupne veze procedure tri
MOV DI,[DI+4] ; u DI ide sadržaj polja pristupne veze procedure dva
; DI ukazuje na kontrolnu vezu procedure jedan pa se uz poznati
; pomeraj pristupa promenljivoj local1
MOV [DI-2],10
```

[1]Dovoljna su i dva pokazivača jer se lokalnim promenljivama može pristupiti koristeći registar BP. Treći pokazivač koristi se pri uspostavljanju displeja prilikom sekvence pozivanja.

Algoritam za postavljanje displeja

- Pri svakom pozivu i povratku ažuriraju se registri displeja da odgovaraju trenutno aktivnoj proceduri (onoj čiji je AZ na vrhu steka).
- Procedura na nivou ugneždavanja N koristi N registara displeja.
- U svakoj situaciji dovoljno je ažurirati jedan registar displeja (to je onaj koji odgovara nivou ugeždavanja N na kome se nalazi pozvana procedura P):
  - pri pozivu, stari sadržaj displej registra N čuva se u akt. zapisu procedure P
  - u displej registar N stavlja se pokazivač na upravo formirani AZ(P)
  - pri povratku iz P, sačuvana vrednost displej registra N restaurira se.

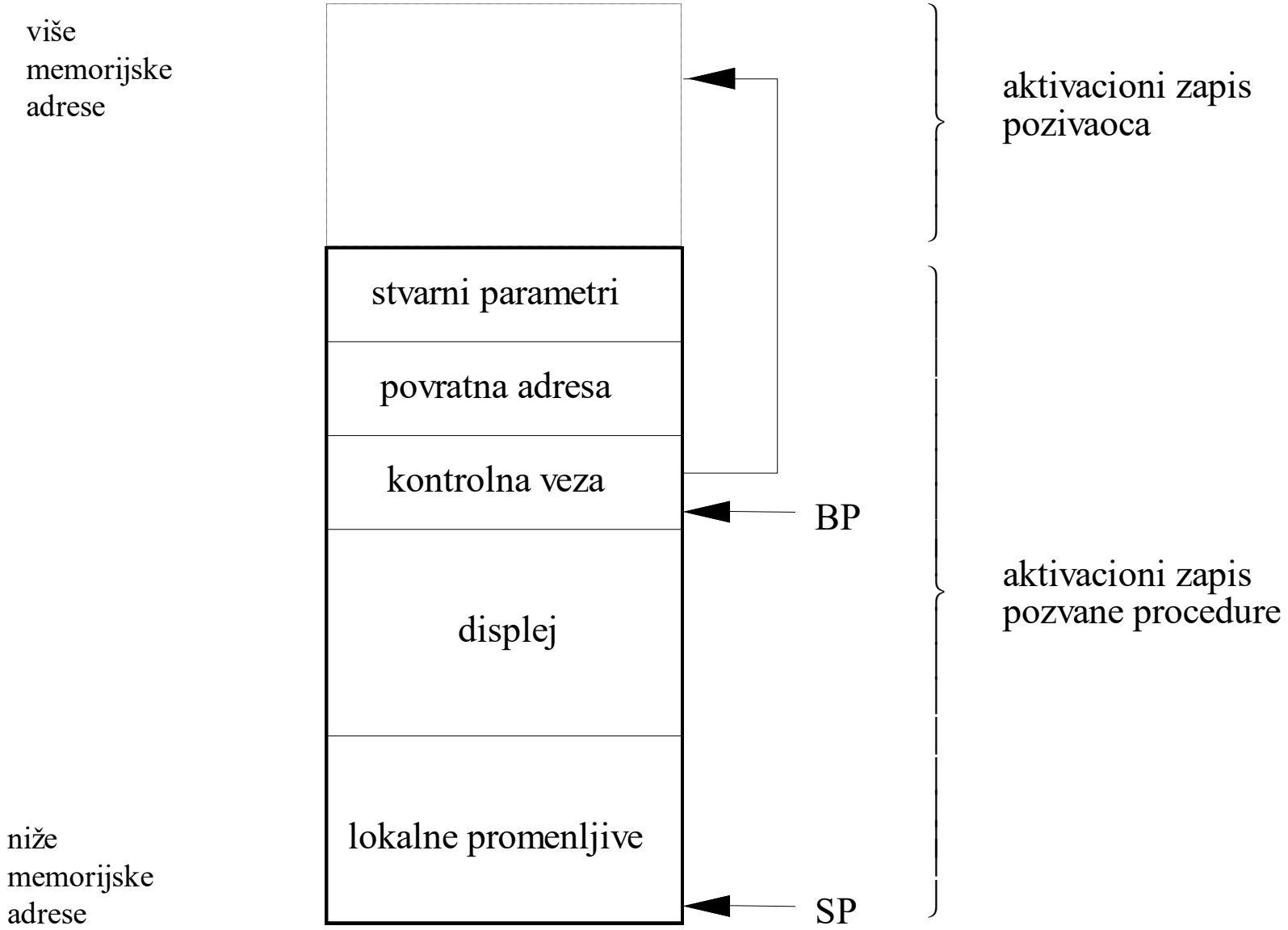
Procesorska podrška višim programskim jezicima

Naslednici procesora 8086 imaju dve mašinske instrukcije za podršku izvršnoj organizaciji memorije kod viših programskih jezika:

- ENTER Size, Level
- LEAVE

Struktura aktivacionog zapisa za ENTER i LEAVE

Na slici je prikazan izgled aktivacionog zapisa koji odgovara ovim naredbama. Za pristup nelokalnim promenljivama koristi se displej alociran u aktivacionom zapisu procedure.



Naredba ENTER Size, Level

- Naredba **ENTER** realizuje deo sekvence pozivanja koji izvršava pozvana procedura.
- Ova naredba ima dva operanda.
  - **Size** (16-bitna veličina) definiše koliko memorije treba alocirati u aktivacionom zapisu za lokalne promenljive.
  - **Level** (8-bitna veličina) predstavlja leksički nivo pozvane procedure.
- Ova naredba izvršava se kao prva naredba pozvane procedure na sledeći način:

1. Sadržaj registra BP ide na stek (čime se definiše polje kontrolne veze)
2. U registar BP upisuje se sadržaj registra SP (čime je BP postavljen da ukazuje na polje kontrolne veze).
3. Ako je Level = 0 preskače se sledeći korak (Level = 0 važi za procedure u C-u kod koga se ne javlja problem pristupa nelokalnim promenljivim).
4. Označimo vrednost Level -1 sa N, a sadržaj polja kontrolne veze sa A. Na stek se smešta N reči koje se kopiraju redom počev od lokacije A-2 ka nižim memorijskim lokacijama. Potom se na stek smešta adresa polja kontrolne veze. Ovime je uspostavljen displej pozvane procedure.
5. Registar SP umanjuje se za vrednost operanda Size čime se alocira prostor za lokalne promenljive pozvane procedure.

Naredba LEAVE

- Naredba **LEAVE** realizuje deo sekvence povratka koji izvršava pozvana procedura.
- Ova naredba neposredno prethodi naredbi RET i izvršava se na sledeći način:
  1. Sadržaj registra BP kopira se u registar SP. Ovime se dealocira sa steka deo aktivacionog zapisa pozivaoca ispod kontrolne veze.
  2. Sa steka se skida sadržaj polja kontrolne veze i smešta u registar BP. Time je BP namešten za pozivaoca, a polje kontrolne veze pozvane procedure je uklonjeno sa steka.

Enter-Leave primer

Odrediti asemblerski ekvivalent datog Pascal programa koristeći predložene instrukcije.

```
program glavni;
        procedure jedan;
        var local1: integer;
                procedure dva;
                        procedure tri;
                        begin {tri}
                                local1:=10
                        end; {tri}
                begin {dva}
                        tri
                end; {dva}
        begin {jedan}
                dva
        end; {jedan}
begin {glavni}
        jedan
end. {glavni}
```

Rešenje

Za dati program, asemblerski ekvivalent uz korišćenje naredbi ENTER i LEAVE bi imao sledeći izgled:

procedura dva

```
; nema lokalnih promenljivih a leksički nivo je jednak 2
ENTER 0,2
; poziv procedure tri
CALL    tri
; sekvenca povratka iz procedure dva
; dealocira se deo aktivacionog zapisa iza polja kontrolne veze
LEAVE
; povratak pozivaocu uz dealociranje polja pristupne veze
; nema stvarnih parametara za dealociranje pa je upotrebljena
; naredba RET bez argumenta
RET
```

procedura tri

```
; nema lokalnih promenljivih a leksički nivo je jednak 3
ENTER 0,3
; prevod naredbe local1 := 10
; u DI se kopira pokazivač koji odgovara leksičkom nivou 1
MOV DI,[BP-02]
; u kativacionom zapisu procedure jedan displej ima samo jedan pokazivač
; promenljiva local1 nalazi se odmah ispod toga pokazivača
MOV [DI-04],10*
; sekvenca povratka
LEAVE
RET
```

* Na 8086 asembleru sintaksa ove instrukcije glasi MOV SS:WORD PTR [DI-2],10
SS: označava da se pristupa segmentu steka (podrazumevana vrednost za registar DI je segment podataka DS),
WORD PTR označava da se prenosti jedna reč, a ne jedan bajt.

procedura jedan

```
; rezervišu se dva bajta za promenljivu local1
; leksički nivo je jednak 1
ENTER 2,1
; poziv procedure dva
CALLdva
; sekvenca povratka
LEAVE
RET
```

glavni program - sekvenca poziva

```
CALLjedan
```