Prijenos parametara i vraćanje rezultata

Primjer

Kao prethodna dva primjera, ali se parametri i rezultat prenose stogom.

```
;;;;; GLAVNI PROGRAM
      LOADL  R7, 10000  ; POSTAVI SP

      LOAD   R0, (PRVI) ; POSTAVI PARAMETRE
      PUSH   R0
      LOAD   R0, (DRUGI)
      PUSH   R0

      CALL   NILI       ; POZOVI POTPROGRAM
      POP    R0         ; SPREMI REZULTAT
      STORE  R0, (REZ)
      HALT                                ➜
```


Prijenos parametara i vraćanje rezultata

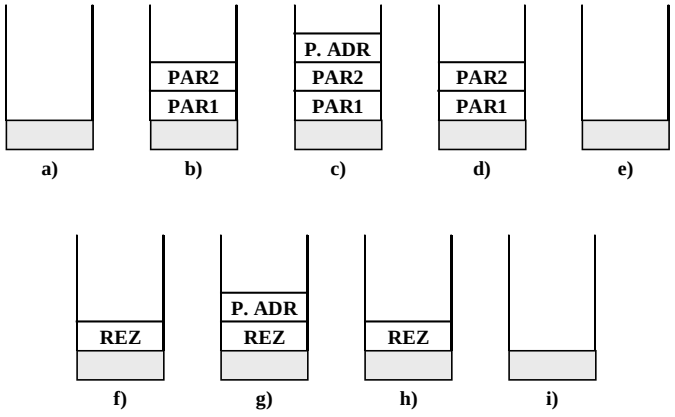
```
;;;;;;;;; PODATCI I MJESTO ZA REZULTAT
PRVI     `DW  81282C34
DRUGI    `DW  29A82855
REZ      `DW  0

;;;;;;;;; POTPROGRAM
NILI      POP    R3    ; povratnu adresu u R3

          POP    R0    ; prvi parametar
          POP    R1    ; drugi parametar
          OR     R0, R1, R0
          COMPL  R0, R0
          PUSH   R0    ; stavi rezultat na stog

          PUSH   R3    ; vrati povratnu adresu
          RET
```


Prijenos parametara i vraćanje rezultata



Prijenos parametara i vraćanje rezultata

Drugačije rješenje - parametre uklanja glavni program, a potprogram im izravno pristupa.

```
;;;;; GLAVNI PROGRAM
      LOADL  R7, 10000  ; POSTAVI SP
      LOAD   R0, (PRVI) ; POSTAVI PARAMETRE
      PUSH   R0
      LOAD   R0, (DRUGI)
      PUSH   R0
      CALL   NILI       ; POZOVI POTPROGRAM
      POP    R2         ; SPREMI REZULTAT
      STORE  R2, (REZ)
      ADD    2, SP, SP
      HALT                                  ➜
```


Prijenos parametara i vraćanje rezultata

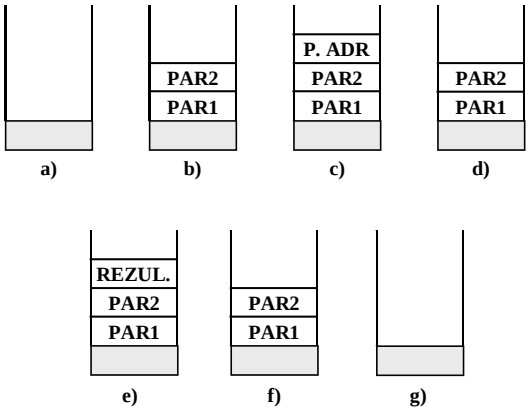
```
;;;;;;;;; PODATCI I MJESTO ZA REZULTAT
PRVI     `DW  81282C34
DRUGI    `DW  29A82855
REZ      `DW  0

;;;;;;;;; POTPROGRAM
NILI     LOAD   R0, (SP+1)  ; prvi parametar
         LOAD   R1  (SP+2)  ; drugi parametar

         OR     R0, R1, R0
         COMPL  R0, R0

         POP    R1    ; povratna adr. u R1
         PUSH   R0    ; rezultat na stog
         JP     (R1)  ; povratak
```


Prijenos parametara i vraćanje rezultata



Prijenos parametara i vraćanje rezultata

Parametri se prenose stogom, a rezultat se vraća registrom R0. Parametre uklanja glavni program. Vrijednosti registara iz glavnog programa moraju ostati sačuvane.

```
;;;;; GLAVNI PROGRAM
      LOADL  R7, 10000  ; POSTAVI SP
      LOAD   R0, (PRVI) ; POSTAVI PARAMETRE
      PUSH   R0
      LOAD   R0, (DRUGI)
      PUSH   R0
      CALL   NILI       ; POZOVI POTPROGRAM
      STORE  R0, (REZ)  ; SPREMI REZULTAT
      ADD    2, SP, SP
      HALT
```


Prijenos parametara i vraćanje rezultata

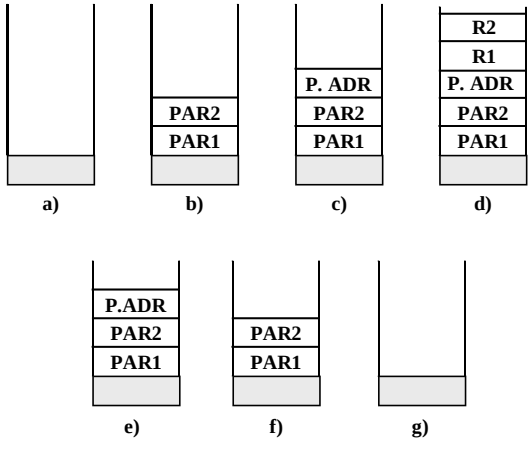
```
NILI      PUSH   R1          ; spremi R1
          PUSH   R2          ; spremi R2

          LOAD   R1, (SP+3)    ; tijelo
          LOAD   R2  (SP+4)    ; potprograma
          OR     R1, R2, R0    ; mijenja
          COMPL  R0, R0        ; R1 i R2

          POP    R2          ; obnovi R2
          POP    R1          ; obnovi R1

          RET                ; povratak
```


Prijenos parametara i vraćanje rezultata



Rekurzivni poziv potprograma

Primjer

Napisati rekurzivni potprogram z aračunanje niza Fibonaccijevih brojeva:

Fib(1) = Fib(2) = 1
Fib(n) = Fib(n-1) + Fib(n-2) za n=3,4, ...

Parametar se unosi stogom, a rezultat se vraća preko R0.

Glavni program treba izračunati *Fib(8)*.

Rješenje je napravljeno tako da pozivatelj potprograma uklanja parametre sa stoga.

➜

Rekurzivni poziv potprograma

Rješenje u C-u:

```
int fib ( int n ) {

   int x, y;

   if ( n == 1  ||  n == 2 )
      return ( 1 );

   y = fib ( n-1 );
   x = fib ( n-2 );
   return ( x+y );
}
```

`return(fib(n-1)+fib(n-2));`

➜

Rekurzivni poziv potprograma

```
GLAVNI LOADL  SP, 10000  ; inicijalizacija stoga

       LOAD   R0, (N)    ;; dohvati parametar iz memorije
       PUSH   R0         ;; i stavi ga na stog

       CALL   FIB        ; poziv potprograma

       STORE  R0, (REZ)  ; spremi rezultat iz R0
       ADD    1, SP, SP  ; ukloni parametar sa stoga

       HALT

N      `DW  8            ; podatak i
REZ    `DW  0            ; mjesto za rezultat
```

➜

Rekurzivni poziv potprograma

```
;;;;;    potprogram

FIB      PUSH   R1          ;; spremi registre
         PUSH   R2          ;; koje potprogram mijenja

         SUB    2, SP, SP   ; rezerviraj mjesto za x i y

;;;;;    if( n==1 || ==2 )  return ( 1 );
         LOADL  R0, 1  ; stavi povratnu vrijednost u
                       ; R0 za slučaj izlaska iz
                       ; potprograma

         LOAD   R1, (SP+5)  ; dohvati parametar n
         CMP    R1, 1
         RET_EQ             ; povratak ako je n==1
         CMP    R1, 2
         RET_EQ             ; povratak ako je n==2
```

➜

Rekurzivni poziv potprograma

```
;;;;;;  x = fib( n-1 );
        LOAD   R1, (SP+5)  ; dohvati parametar n
        SUBR   R1, 1, R1   ;; izračunaj n-1 i stavi ga na
        PUSH   R1          ;; stog za prvi rekurzivni poziv

        CALL   FIB         ; pozovi fib(n-1)
        STORE  R0, (SP+1)  ; spremi rezultat u x
        ADD    1, SP, SP   ; ukloni n-1 sa stoga

;;;;;   y = fib( n-2 );
        LOAD   R1, (SP+5)  ; dohvati parametar n
        SUBR   R1, 2, R1   ;; izračunaj n-2 i stavi ga na
        PUSH   R1          ;; stog za prvi rekurzivni poziv

        CALL   FIB         ; pozovi fib(n-2)
        STORE  R0, (SP+0)  ; spremi rezultat u y
        ADD    1, SP, SP   ; ukloni n-1 sa stoga2
```

➜

Rekurzivni poziv potprograma

```
;;;;;;  return( x+y );
        LOAD   R1, (SP+0)  ; dohvati y
        LOAD   R2, (SP+1)  ; dohvati x
        ADD    R1, R2, R0  ; izračunaj povratnu vrijednost

        ADD    2, SP, SP   ; ukloni x i y sa stoga

        POP    R2          ;; obnovi sadržaje
        POP    R1          ;; spremljenih registara
        RET                ; povratak iz FIB2
```

Različiti primjeri programiranja

Primjer

Jednostruko povezana sortirana lista ima čvorove koji zauzimaju dvije memorijske lokacije: na prvoj je NBC broj (koji predstavlja vrijednost čvora), a na drugoj je pokazivač na sljedeći čvor liste (tj. adresa sljedećeg čvora). Prvi čvor liste ima specijalno značenje i u njemu se pamti zbroj svih vrijednosti preostalih čvorova liste. Prvi čvor je uvijek prisutan, bez obzira postoje li ostali čvorovi. Zadnji čvor u listi prepoznaje se po NULL-pokazivaču (tj. u lokaciji s adresom sljedećeg čvora upisana je ništica). Pretpostavka je da u zbroju nikada neće doći do prekoračenja opsega.

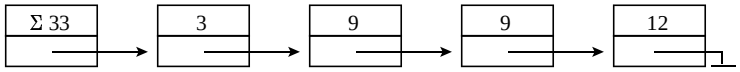
➜

16

Različiti primjeri programiranja

Treba napisati potprogram UBACI koji ubacuje novi čvor u postojeću sortiranu listu tako da ona ostane sortirana. Adresa prvog elementa liste i adresa novog čvora predaju se preko stoga. Povratna vrijednost je novi zbroj iz prvoga čvora liste. Vrijednost se vraća pomoću R0.

➜



17

```
UBACI POP    R0         ; povratna adresa u R0
      POP    R1         ; adresa prvog čvora liste
      POP    R2         ; adresa novog čvora

      LOAD   R6, (R2) ; dohvati vrijednost novog čvora
      MOVE   R1, R3   ; pripremi adresu za 1. korak petlje

TRAZI LOADcc R4, (R3+1) ; dohvati adresu trenutačnog čvora
      JR_Z   NASAO      ; provjeri je li kraj liste

      ; dohvati vrijednost trenutačnog čvora i
      ; usporedi je s vrijednošću novog čvora
      LOAD   R5, (R4)
      CMP    R5, R6

      JR_UGE NASAO   ; ako je trenutačni   od novog =>
                     ; pronađeno je mjesto za ubacivanje

      ; inače treba nastaviti s petljom za traženje
      MOVEcc R4, R3     ; pomakni se na sljedeći čvor
      JR     TRAZI      ; nastavi s traženjem
                                                  ➜
```

18

```
NASAO STORE  R2, (R3+1) ; stavi novi čvor iza prethodnog
      STORE  R4, (R2+1) ; stavi trenutačni čvor iza novog

      ; izračunavanje novog zbroja
      LOAD   R5, (R1)   ; dohvati dosadašnji zbroj
      LOAD   R6, (R2)   ; dohvati vrijednost novog
                        ; elementa

      ADD    R5, R6, R5 ; novi zbroj stavi u R5
      STORE  R5, (R1)   ; spremi je u prvi čvor

      PUSH   R0         ; vrati povratnu adresu na stog
      MOVE   R5, R0     ; postavi povratnu vrijednost u R0
      RET
```


Različiti primjeri programiranja

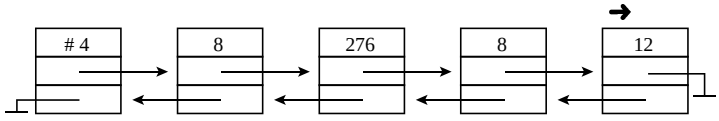
Primjer

U memoriji se nalazi dvostruko povezana nesortirana lista. Svaki čvor zauzima tri memorijske lokacije: na prvoj je NBC broj, na drugoj je pokazivač na sljedeći čvor liste, a na trećoj je pokazivač na prethodni čvor liste. Prvi čvor liste ima specijalno značenje i uvijek je prisutan u listi. U njemu se pamti ukupan broj preostalih čvorova u listi.

➜

Različiti primjeri programiranja

Treba napisati potprogram IZBACI koji prima preko stoga dva parametra: pokazivač na prvi čvor liste i NBC broj (X). Potprogram traži prvo pojavljivanje čvora u kojemu je upisan broj X. Ako ga nađe, izbacuje taj čvor iz liste. Povratna vrijednost je adresa izbačenog čvora ili ništica ako čvor nije pronađen. Vrijednost se vraća pomoću R1. Potprogram ne smije mijenjati sadržaje registara u glavnom programu.



```
IZBACI PUSH   R0          ; spremi registre
       PUSH   R2          ; iz glavnog programa

       LOAD   R2, (SP+3)  ; dohvati broj X
       LOAD   R1, (SP+4)  ; dohvati adresu prvog čvora
       LOADcc R1, (R1+1)  ; pripremi pokazivač za traženje

TRAZI  JR_Z   IZLAZ       ; ako je kraj => čvor nije nađen

       LOAD   R0, (R1+0)  ; dohvati vrijednost čvora
       CMP    R0, R2      ; usporedi je sa X
       JR_EQ  NASAO       ; ako su isti => čvor je nađen

       LOADcc R1, (R1+1)  ; pomakni se na sljedeći čvor
       JR     TRAZI       ; nastavi s traženjem
```

➜


```
NASAO  ;;;; odspoji nađeni čvor iz liste
       LOAD   R0, (R1+2)  ; stavi adresu prethodnog u R0
       LOAD   R2, (R1+1)  ; stavi adresu sljedećeg u R2

       STORE  R2, (R0+1)  ; stavi sljedeći iza prethodnog
       STORE  R0, (R2+2)  ; stavi prethodni ispred sljedeć.

       CLEAR  R0          ; odspoji pokazivače
       STORE  R0, (R1+1)  ; u čvoru kojeg
       STORE  R0, (R1+2)  ; izbacuješ iz liste

       ;;;; smanji brojač čvorova u 1. čvoru liste
       LOAD   R0, (SP+4)  ; dohvati adresu 1. čvora
       LOAD   R2, (R0+0)  ; dohvati broj čvorova liste
       DEC    R2          ; smanji broj čvorova
       STORE  R2, (R0+0)  ; upiši ga natrag u 1. čvor

IZLAZ  POP    R2          ; obnovi registre
       POP    R0          ; iz glavnog programa
       RET
```