Programiranje procesora FRISC

Potprogrami

Potprogrami

- Namjena potprograma u asemblerskom programiranju ista je kao i u višim programskim jezicima.

- Čemu služi potprogram znate već od prije:
  - bolja modularnost i organizacija programa
  - lakše programiranje
  - povećana čitljivost
  - lakše održavanje programa
  - manje zauzeće memorije

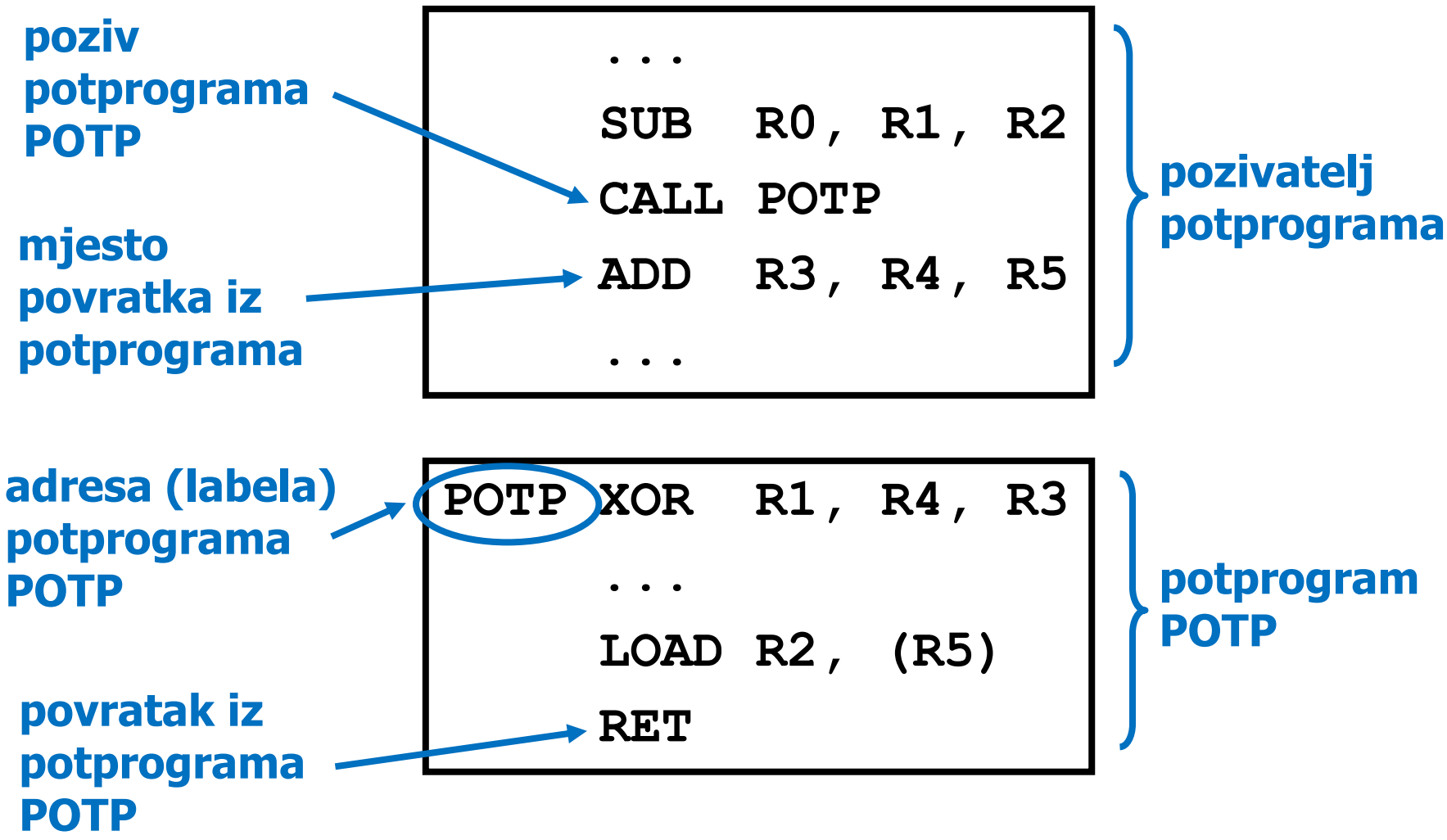
Potprogrami

- Kao i u višim programskim jezicima, moramo znati kako se koriste potprogrami, ali u asemblerskom programiranju moramo dodatno znati i neke detalje "niže razine"

- Trebamo znati:
  - Kako se potprogram poziva i kako se vraćamo iz potprograma
  - Gdje se sprema povratna adresa
  - Kako se prenose parametri i vraćaju rezultati iz potprograma
  - Gdje se čuvaju lokalne varijable potprograma
  - Kako se ostvaruje da stanja registara iz glavnog programa ne budu promijenjena za vrijeme potprograma

- U višim programskim jezicima, o ovim stvarima uglavnom ne moramo voditi računa jer se o njima brine prevoditelj (koji stvara asemblersku verziju našeg programa)

Potprogrami



Potprogrami - CALL i RET

- **Podsjetnik:** kako rade naredbe CALL i RET?
  - Naredba: **`CALL adresa_potprograma`**
    - sprema povratnu adresu iz PC-a na stog
      - to je adresa naredbe IZA naredbe CALL, jer za vrijeme izvođenja naredbe CALL, PC već pokazuje na sljedeću naredbu, tj. sadrži povratnu adresu
    - skače na zadanu adresu **`adresa_potprograma`**
  - Naredba: **`RET`**
    - uzima podatak (povratnu adresu) sa stoga
    - skače na tu povratnu adresu

Potprogrami - CALL i RET

- **Gdje se u memoriji nalazi stog ???**

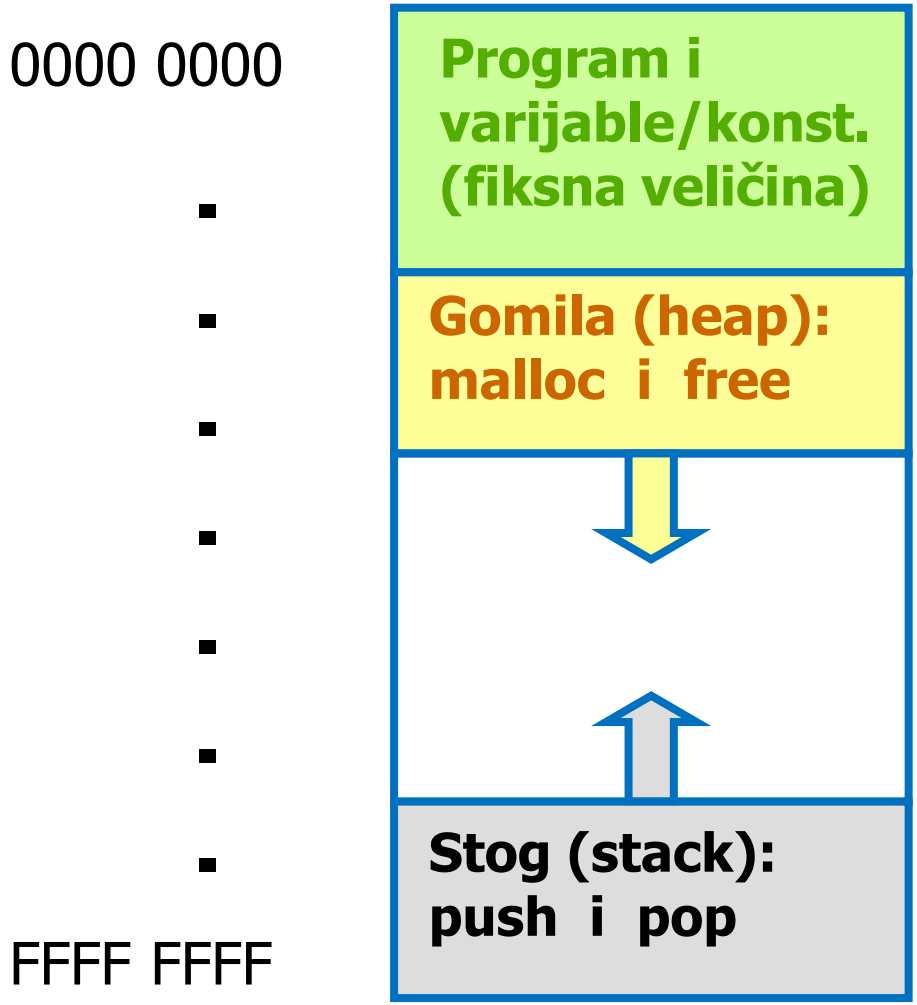
- Položaj stoga u memoriji definiramo programski na početku glavnog programa gdje **MORAMO inicijalizirati pokazivač stoga** (SP)

- Ako to zaboravimo učiniti, doći će do greške u programu (u ATLAS-u će vjerojatno javiti grešku "čitanje sa sabirnice u stanju High Z")

Potprogrami - stog

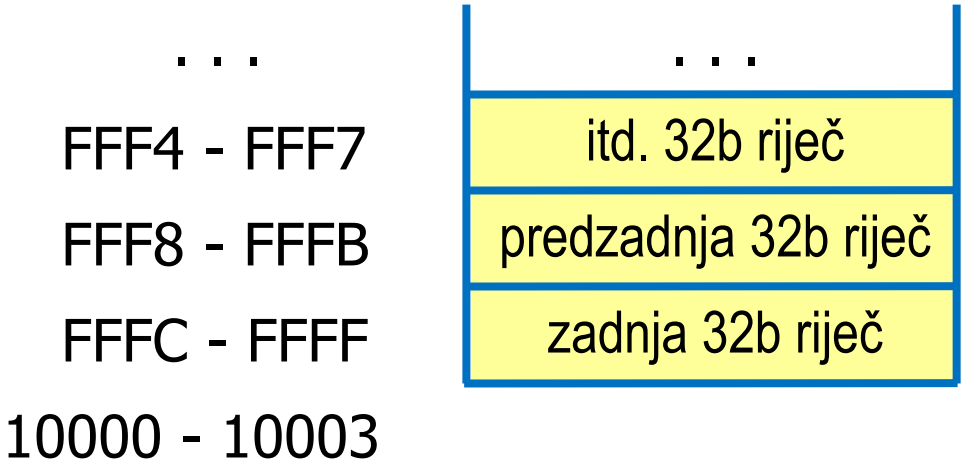
Tipična organizacija memorije:

- Na najnižim adresama se nalazi program i varijable/konstante.
- Iza toga se nalazi područje gomile (heap). Gomila dinamički raste prema najvišim adresama. (*nema veze sa strukturom podataka koja se također naziva heap - iz "Algoritama i struktura podataka"*).
- Na najvišim adresama se nalazi stog koji raste prema nižim adresama.

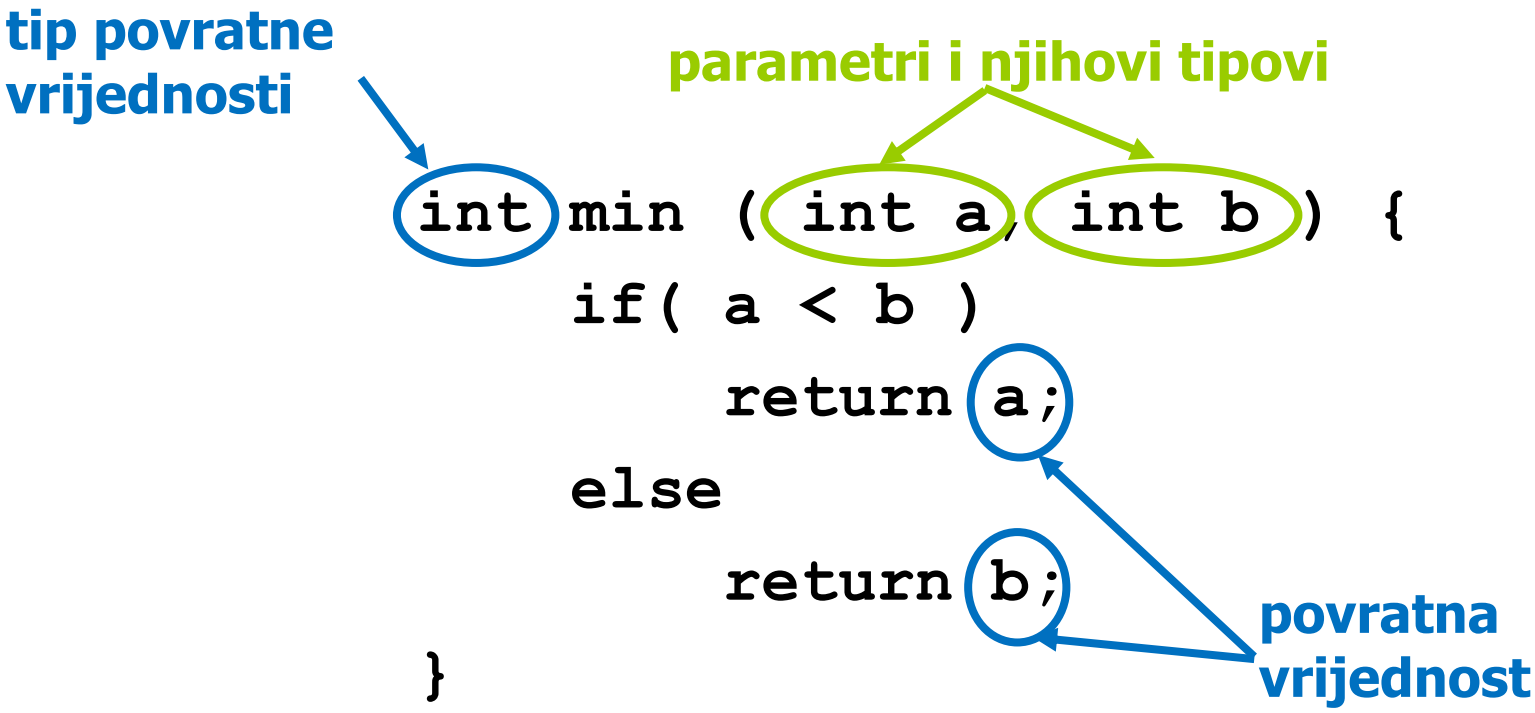


Potprogrami - stog

- Mi ćemo pretpostaviti da je na FRISC spojena memorija od 64 KB. Najviša adresa (8-bitne memorijske lokacije) u memoriji od 64 KB je FFFF
- Na stog se uvijek stavljaju 32-bitni podatci pa im adrese moraju biti djeljive sa 4
- FFFC je adresa 32-bitnog podatka koji je "na zadnjoj 32-bitnoj lokaciji", tj. "na najvišim lokacijama memorije"
- Budući da SP pokazuje na zadnji podatak na stogu (podatak koji na početku još ne postoji), onda SP inicijaliziramo na jednu 32-bitnu riječ dalje: 4+FFFC = 10000



Potprogrami - Parametri i rezultati



Potprogrami - Parametri i rezultati

- U višim programskim jezicima parametri i povratna vrijednost imaju tipove koje provjerava prevoditelj
- U asembleru nema nikakvih provjera ni tipova:
  - Dužnost je programera da se brine kakve podatke šalje u potprogram i kakve rezultate prima iz potprograma
  - Velika fleksibilnost u programiranju
  - Velika mogućnost greške u programu
- Terminologija:
  - parametar (formalni parametar) - "varijabla" u potprogramu preko koje potprogram prima vrijednosti od pozivatelja
  - argument (stvarni parametar) - stvarna vrijednost koju pozivatelj šalje potprogramu preko njegovih parametara

Potprogrami - Parametri i rezultati

- Tri najčešća načina slanja parametara i vraćanja povratne vrijednosti:
  - **Pomoću registara**
  - **Pomoću fiksnih memorijskih lokacija**
  - **Pomoću stoga**

- Teorijski je moguće slobodno kombinirati više različitih načina prijenosa u jednom potprogramu. Na primjer:
  - prvi parametar prenosi se registrom, drugi stogom, a ostali pomoću fiksnih lokacija
  - prvi rezultat se vraća stogom, a drugi fiksnom lokacijom

Prijenos registrima

Potprogrami - Prijenos registrima

- Za svaki potprogram zasebno, propiše se preko kojih registara prima podatke i preko kojih vraća rezultate
- Pozivatelj potprograma (glavni program ili bilo koji drugi potprogram) prije poziva mora staviti argumente u zadane registre
- Potprogram polazi od pretpostavke da su u zadanim registrima argumenti i koristi ih pri izračunavanju rezultata
- Potprogram mora staviti rezultat u zadane registre
- Pozivatelj, nakon povratka iz potrograma, pretpostavlja da se u zadanim registrima nalaze rezultati i koristi ih

Potprogrami - Prijenos registrima

Primjer:

Napisati potprogram koji izračunava logičku operaciju NILI između registara R0 i R1 i vraća rezultat registrom R2. Glavni program iz memorije učitava dva podatka za koje računa NILI (pomoću potprograma) i rezultat sprema natrag u memoriju.

Rješenje:

(na sljedećem slajdu)

>>>>

<<<<

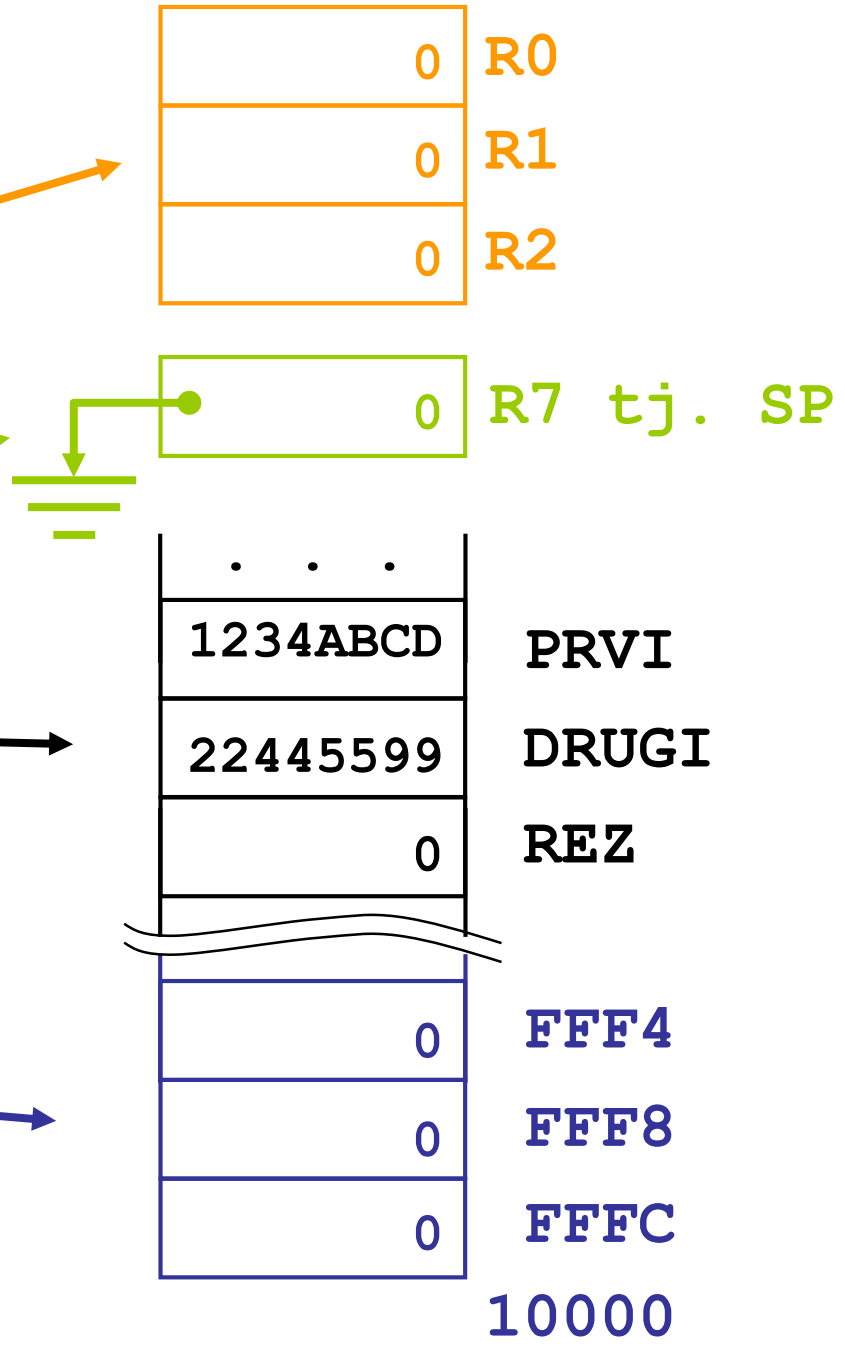
Pratimo stanje sljedećih registara/lokacija:

tri registra opće namjene koje koristimo

pokazivač stoga

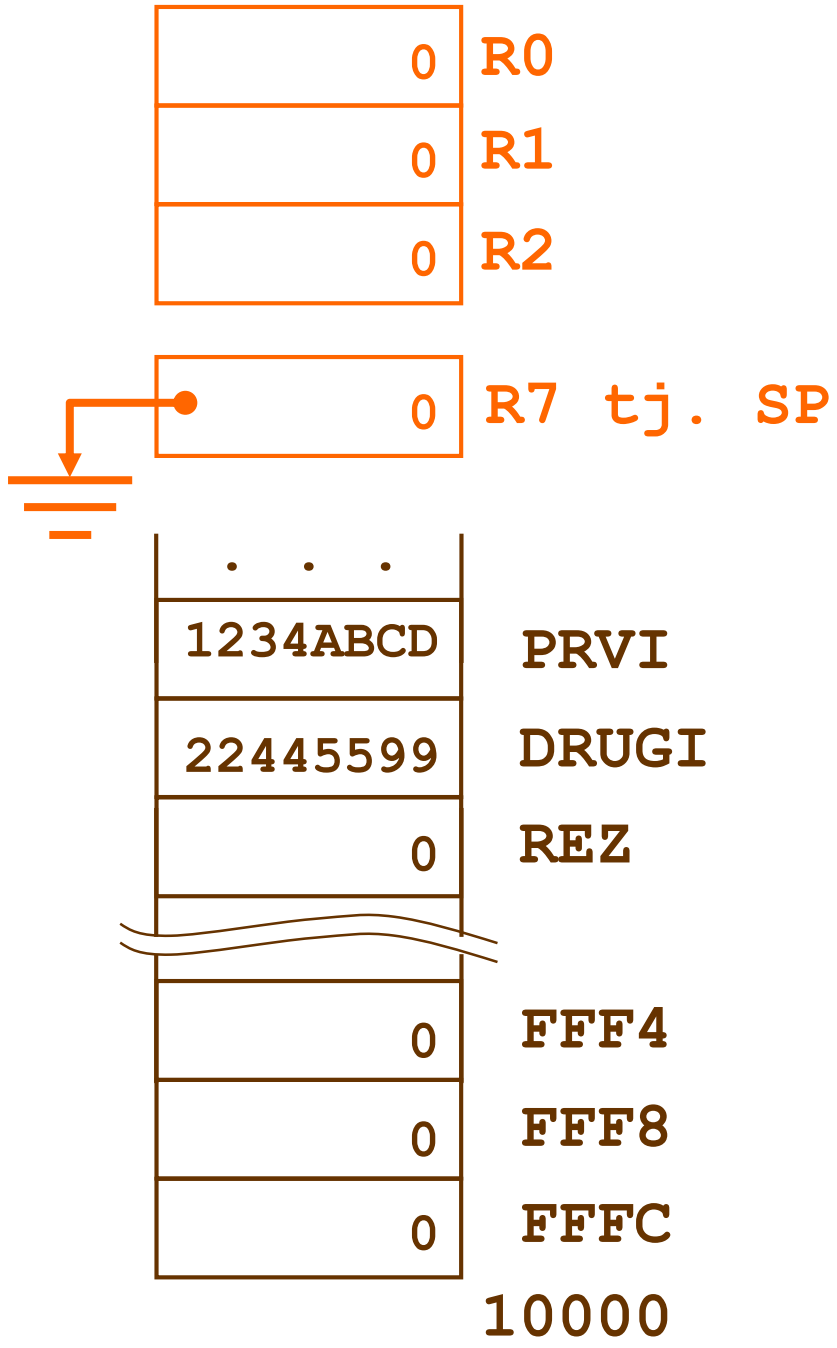
dio memorije s podatcima

dio memorije u kojem je stog



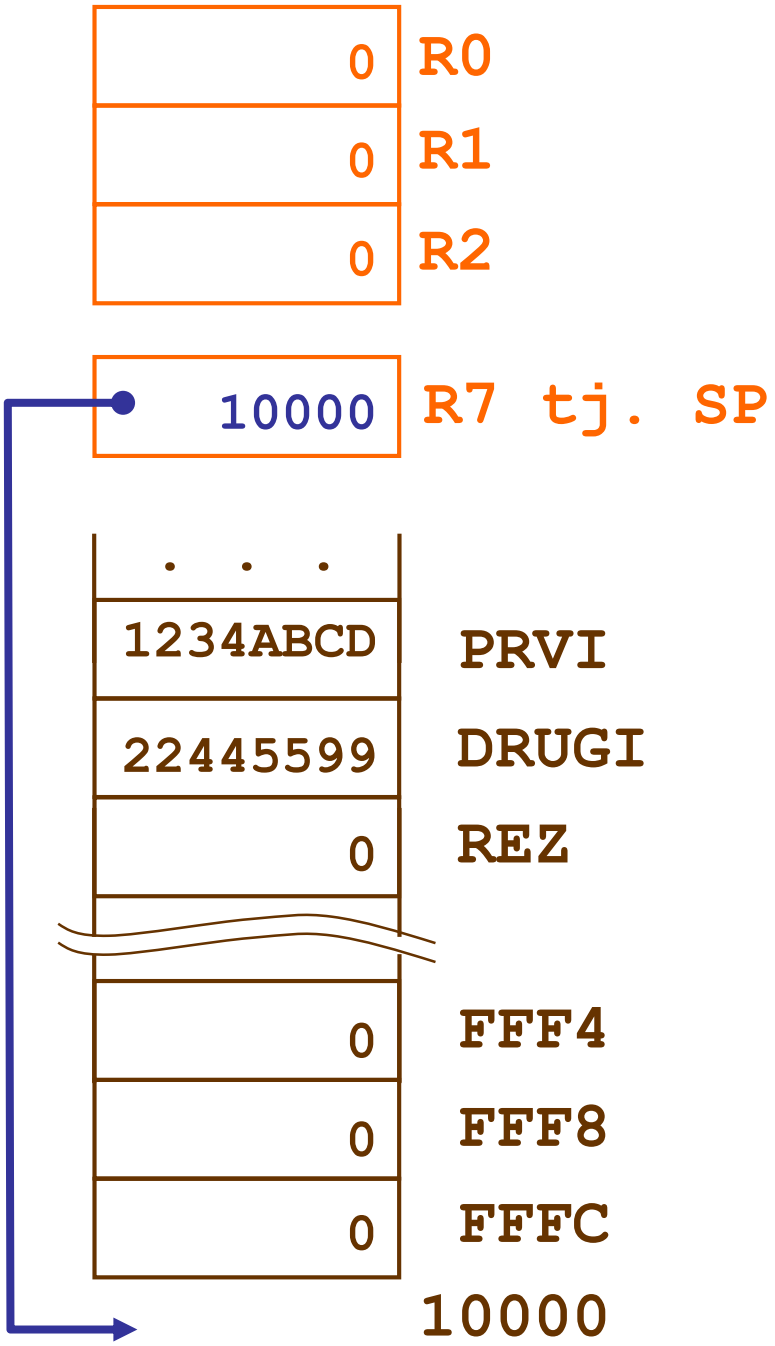
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT


NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



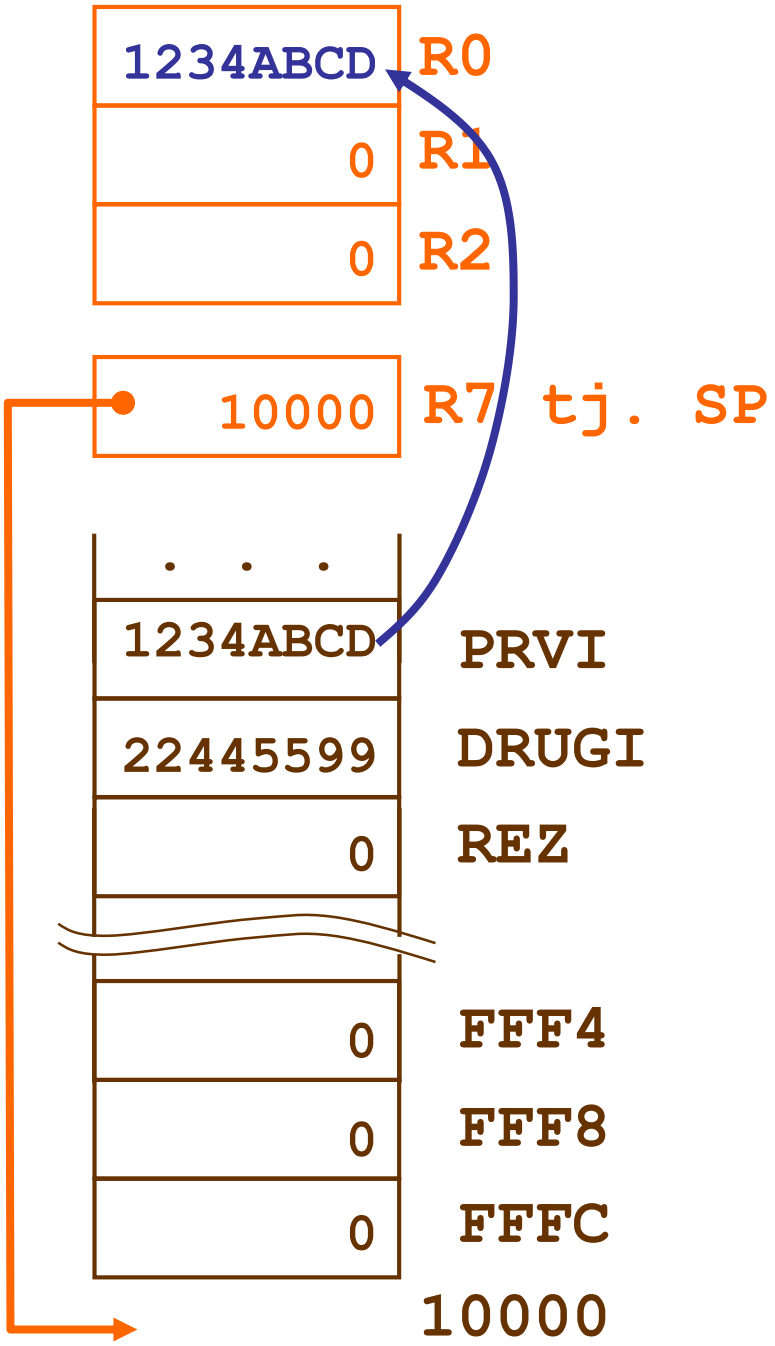
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT


NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



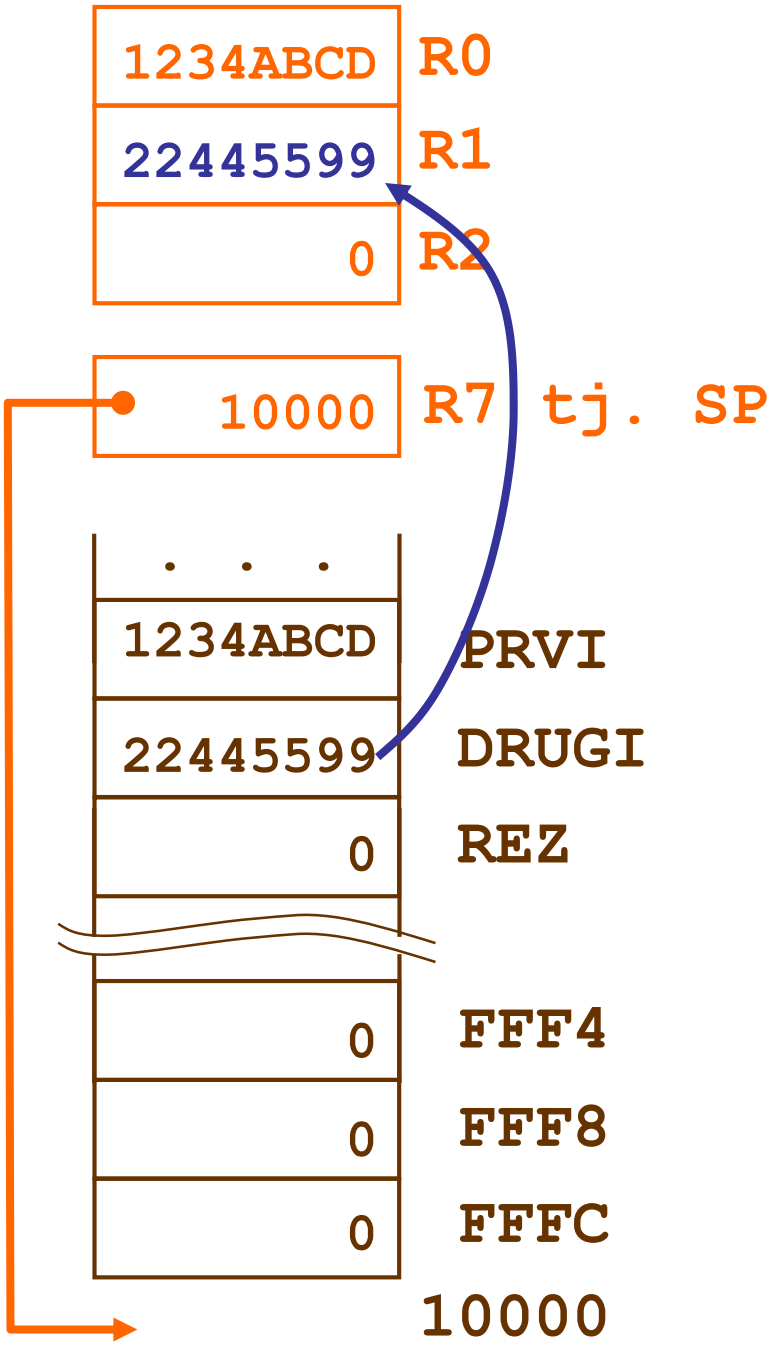
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT



NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



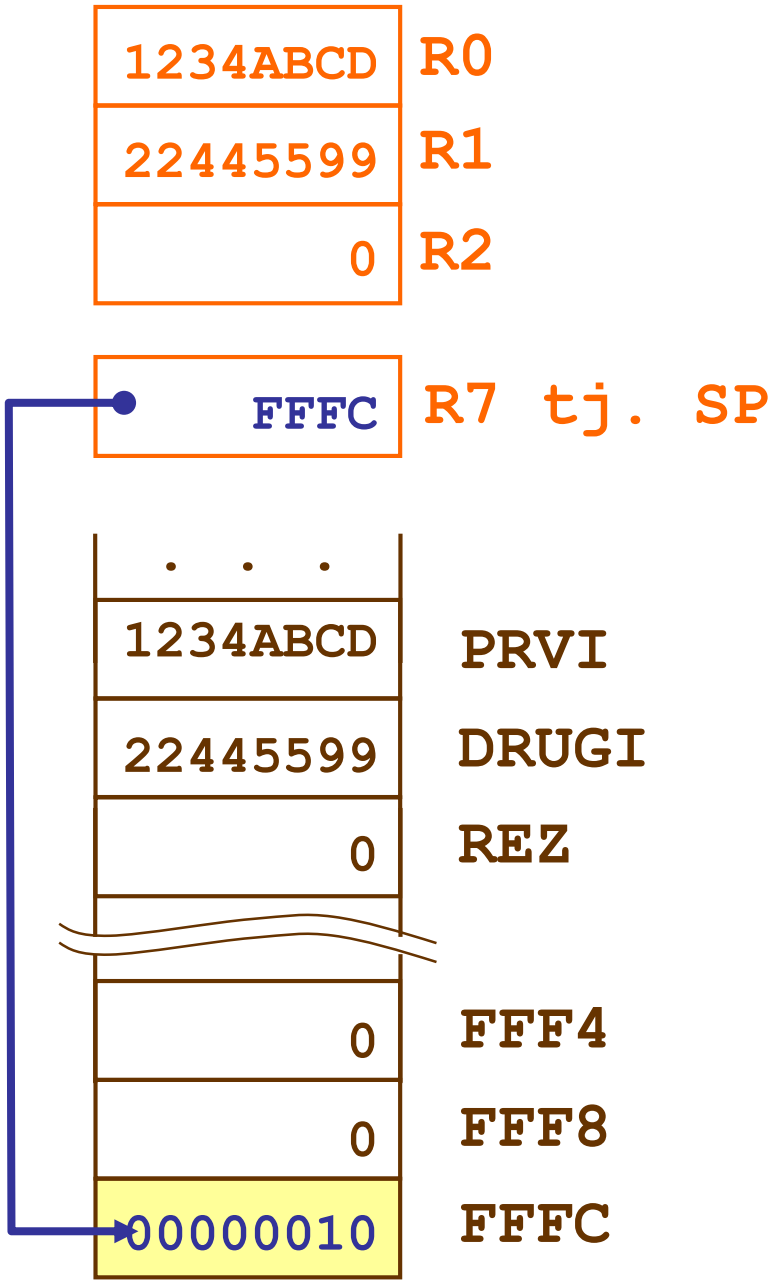
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT



NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



<<<< Izvođenje programa:

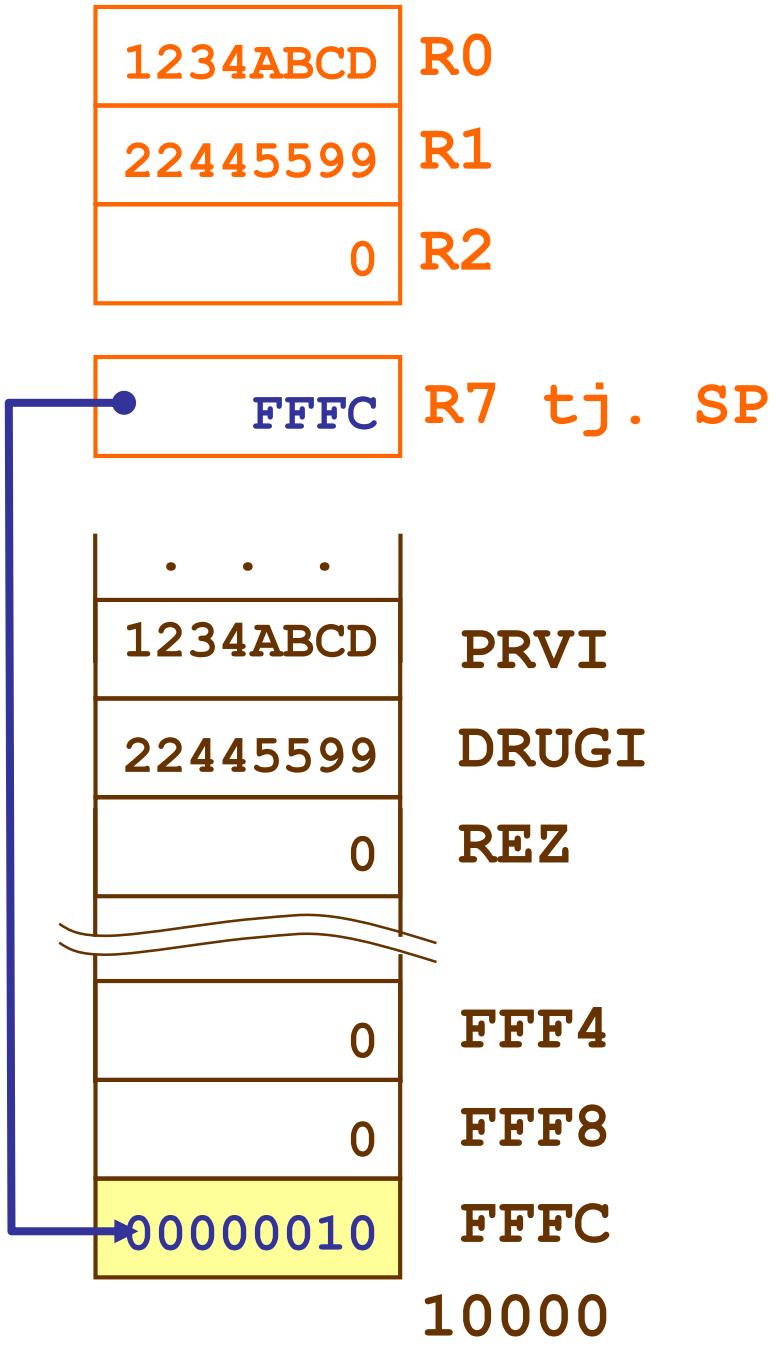
```
GLAVNI  MOVE   10000, SP        0

        LOAD   R0, (PRVI)       4
        LOAD   R1, (DRUGI)      8
        CALL   NILI             C
        STORE  R2, (REZ)       10

        HALT                   14



NILI    OR   R0, R1, R2
        XOR  R2, -1, R2
        RET

PRVI    DW   1234ABCD
DRUGI   DW   22445599
REZ     DW   0
```



00000010 je adresa naredbe STORE

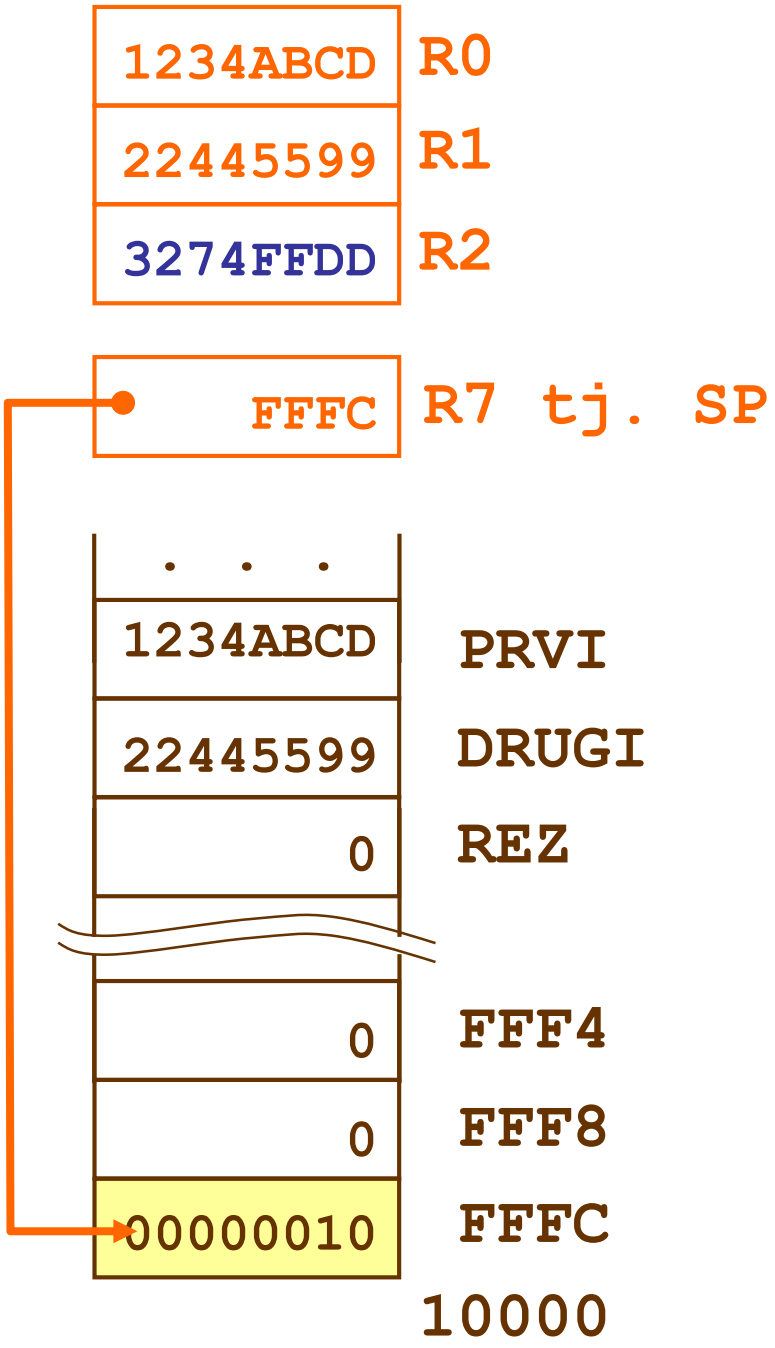
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT



NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



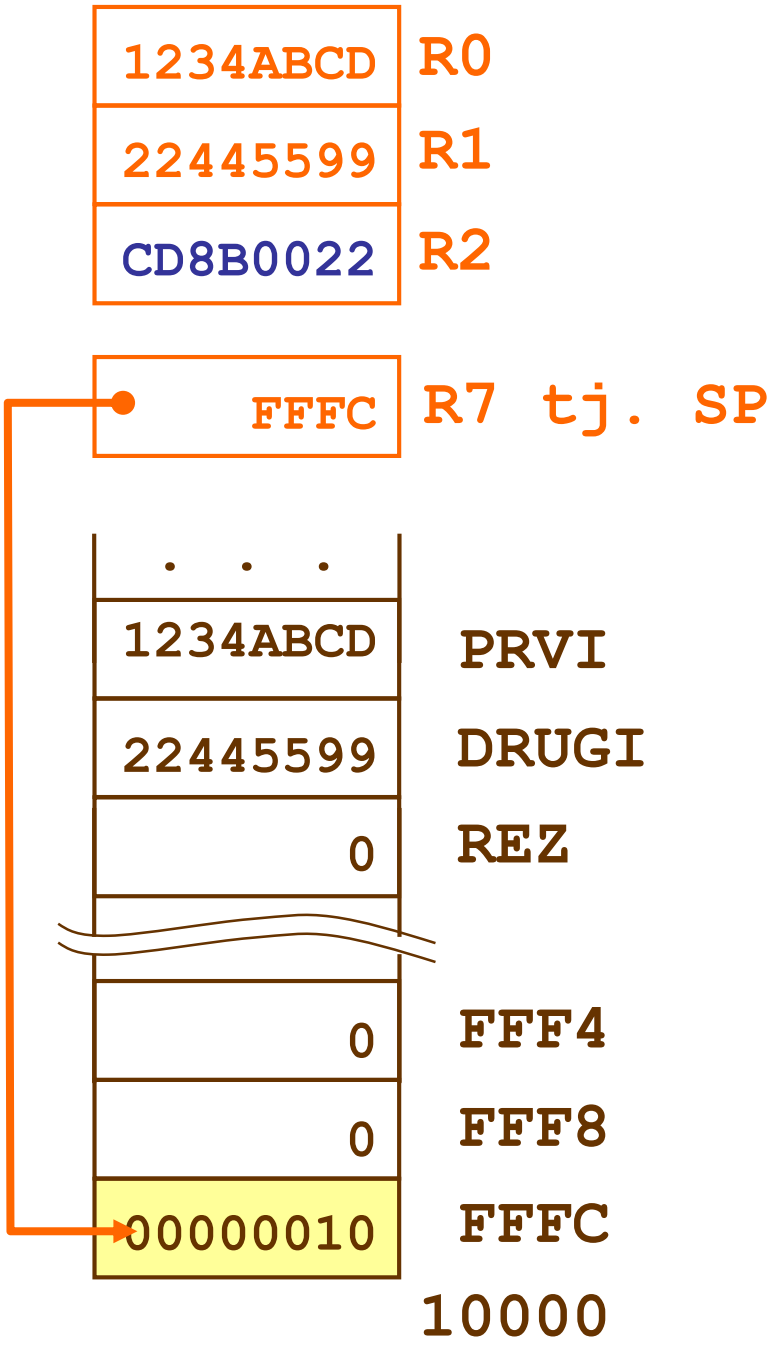
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT


NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



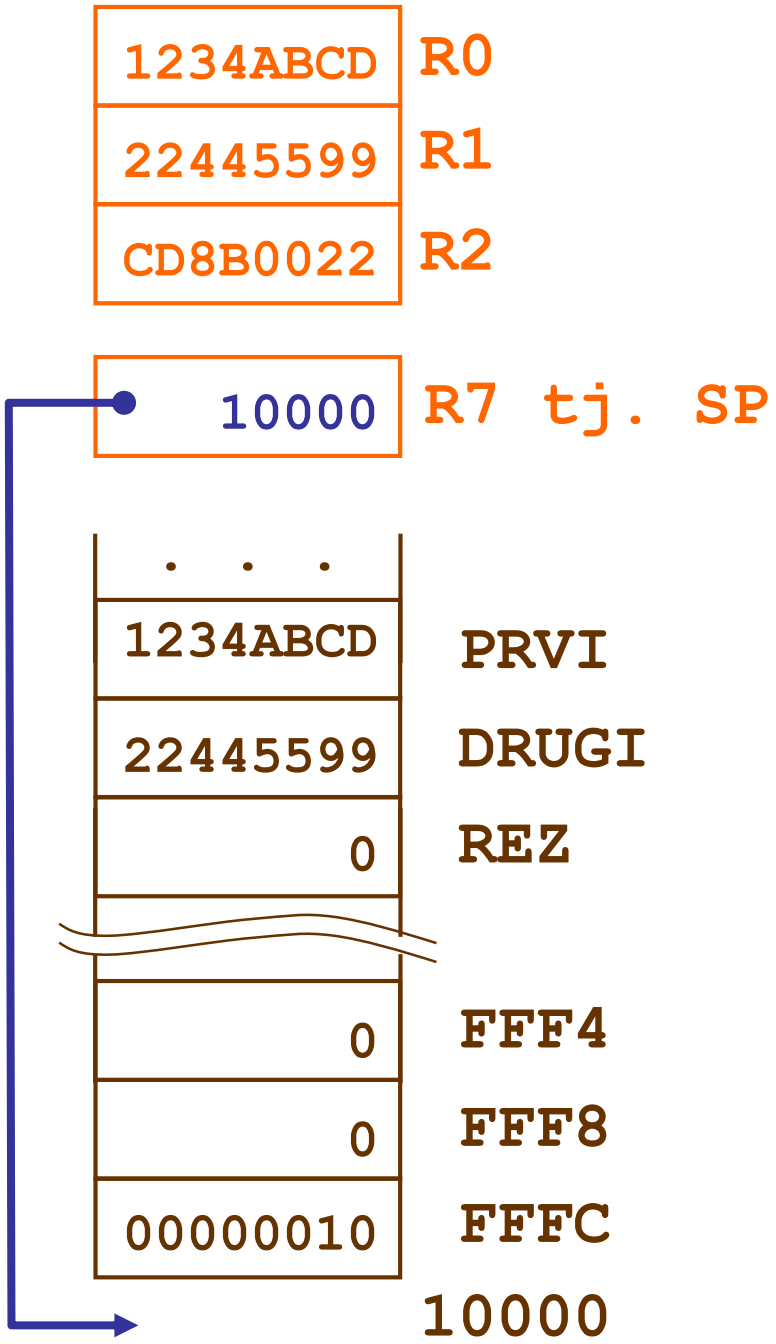
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT


NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



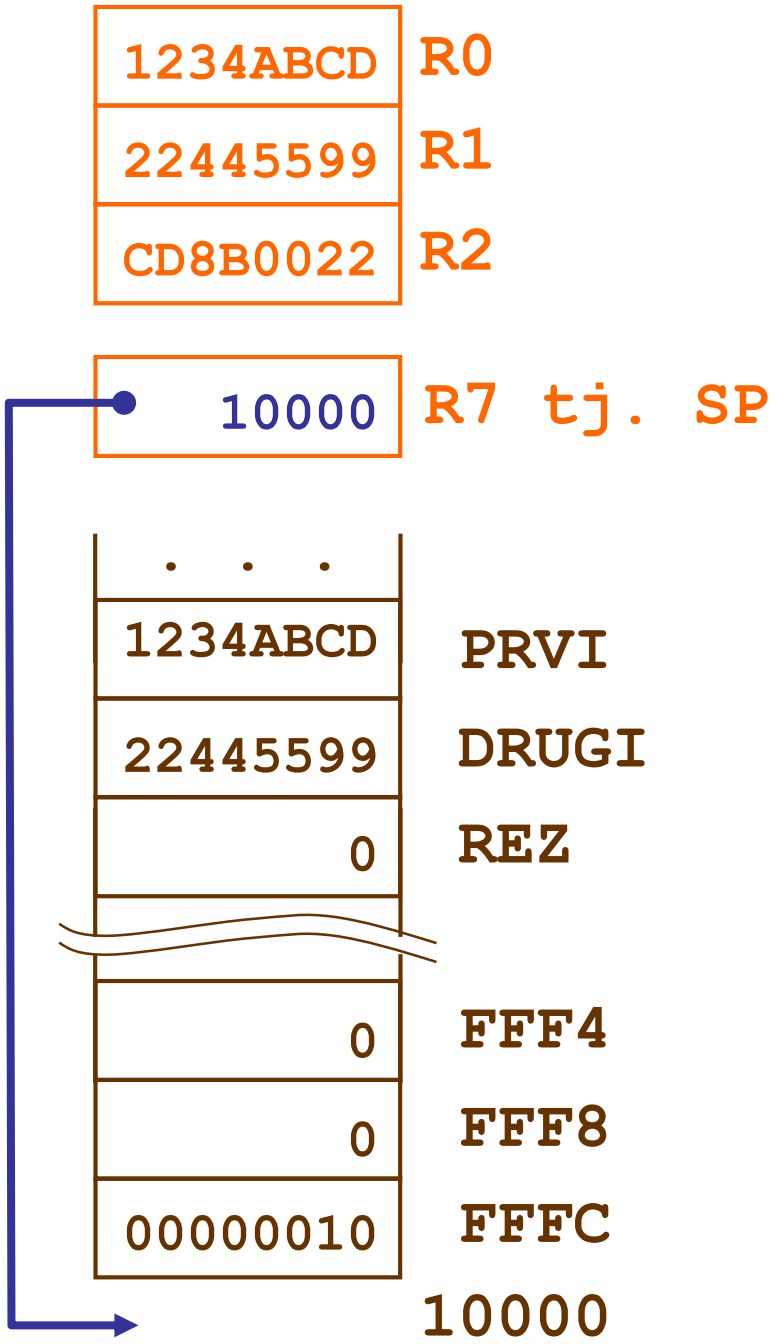
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT



NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



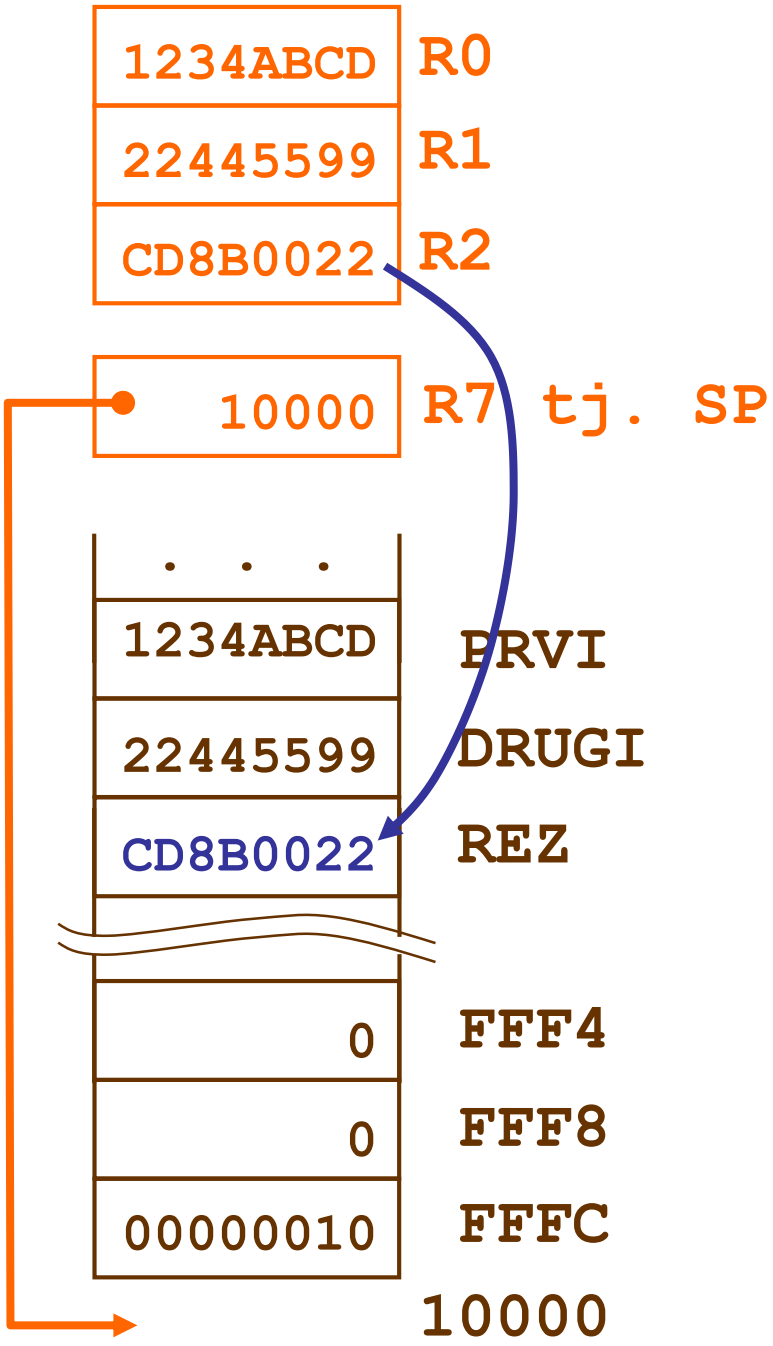
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT



NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



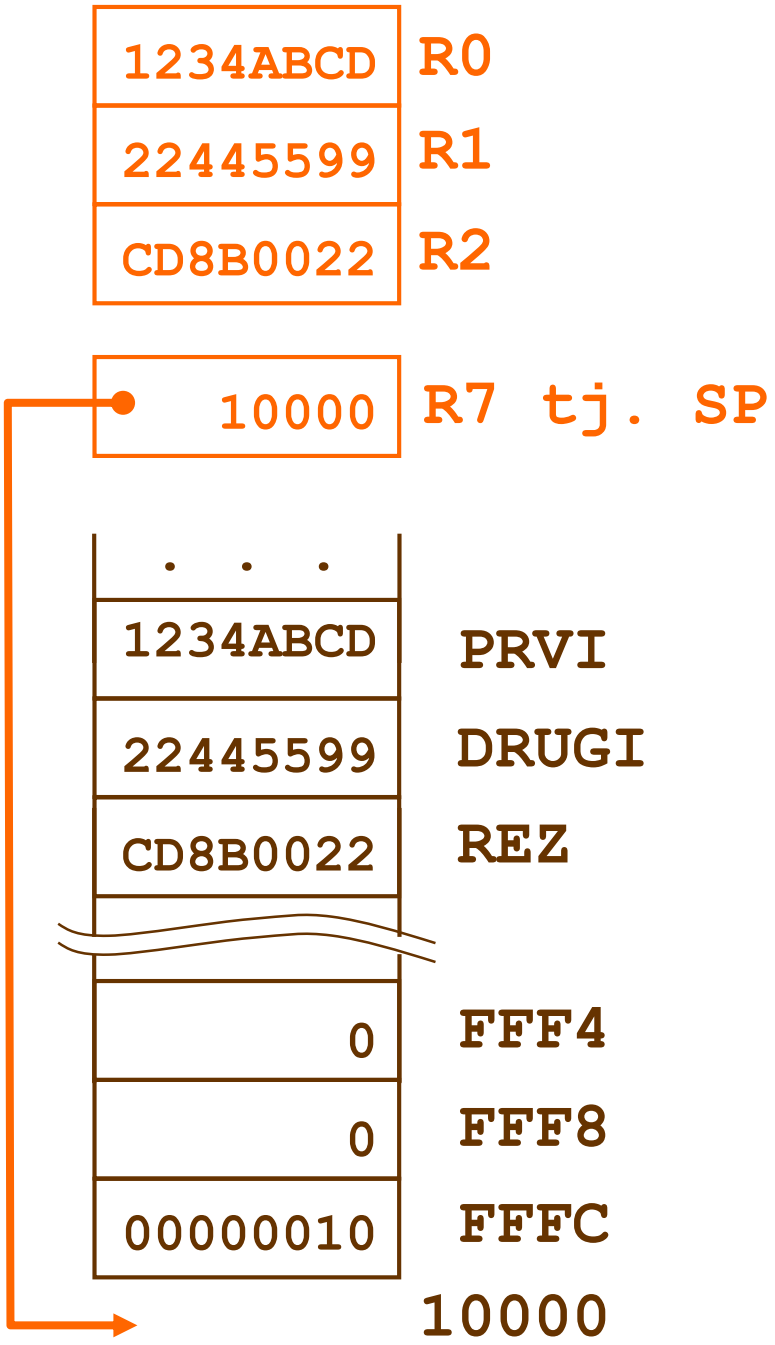
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT


NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



<<<< Izvođenje programa:

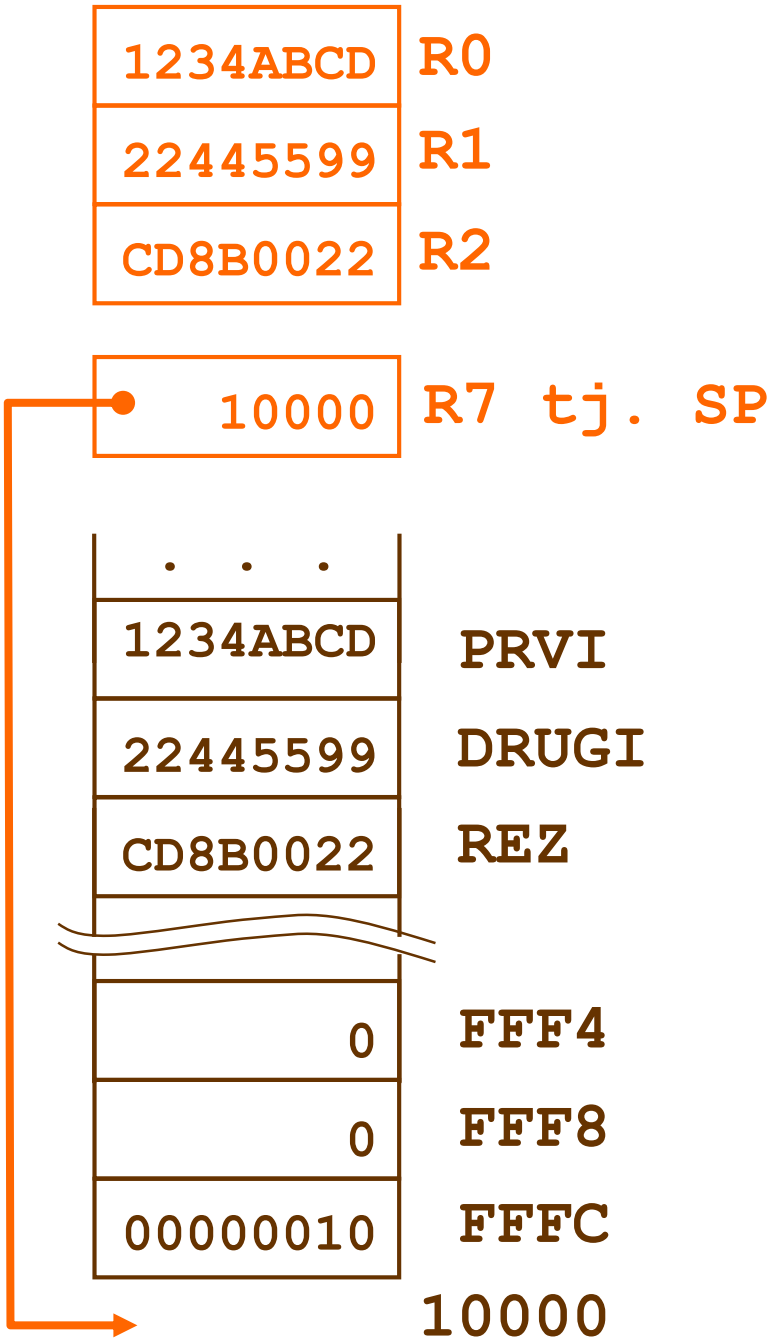
```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       LOAD  R1, (DRUGI)
       CALL  NILI
       STORE R2, (REZ)

       HALT

NILI   OR  R0, R1, R2
       XOR R2, -1, R2
       RET

PRVI   DW  1234ABCD
DRUGI  DW  22445599
REZ    DW  0
```



Komentar:

Treba uočiti da ovaj potprogram mijenja samo sadržaj registra R2 (ne računajući R7). Međutim, R2 je registar preko kojeg se vraća vrijednost, tj. glavni program očekuje da će taj registar biti promijenjen nakon povratka iz potprograma. Zato glavni program prije poziva potprograma ne smije imati u R2 spremljen nikakav podatak koji bi mu još mogao zatrebati.

Registre R0 i R1 promijenio je glavni program, jer je preko njih slao parametre. Zato je za pretpostaviti da glavni program nije imao nikakve korisne podatke u R0 i R1 prije upisa parametara u njih.



<<<< (kompletan listing programa s komentarima)

```
        ; glavni program
GLAVNI  MOVE  10000, SP    ;važno: inicijaliziraj SP !!!
        LOAD  R0, (PRVI)   ;vrijednost u prvi parametar
        LOAD  R1, (DRUGI)  ;vrijednost u drugi parametar
        CALL  NILI         ;poziv potprograma
        STORE R2, (REZ)    ;spremanje rezultata
        HALT

        ; potprogram NILI
NILI    OR  R0, R1, R2   ; koriste se parametri R0 i R1
        XOR R2, -1, R2   ; rezultat se upisuje u R2
        RET              ; povratak iz potprograma

        ; podatci i mjesto za rezultat
PRVI    DW  1234ABCD
DRUGI   DW  22445599
REZ     DW  0
```

Potprogrami - Prijenos registrima

- **Prijenos registrima**
  - Prednosti:
    - Brz prijenos (rad s registrima je najbrži)
    - Jednostavan prijenos (sa stajališta programiranja)
  - Nedostatci:
    - Može se prenijeti ograničen broj argumenata (najviše 7)
    - Registri obično čuvaju međurezultate i nisu slobodni za korištenje kao parametri
    - Nije moguće rekurzivno pozivanje potprograma

Prijenos fiksnim lokacijama

Potprogrami - Prijenos fiksnim lok.

- Pod fiksnim lokacijama misli se na memorijske lokacije čija adresa je unaprijed zadana i ne mijenja se tijekom programa
- Za svaki potprogram zasebno, propiše se preko kojih fiksnih memorijskih lokacija prima podatke i preko kojih vraća rezultate
- Pozivatelj prije poziva mora staviti argumente u zadane lokacije
- Potprogram polazi od pretpostavke da su u zadanim lokacijama argumenti i koristi ih pri izračunavanju rezultata
- Potprogram mora staviti rezultat u zadane lokacije
- Pozivatelj, nakon povratka iz potrograma, pretpostavlja da se u zadanim lokacijama nalaze rezultati i koristi ih

Potprogrami - Prijenos fiksnim lokacijama

Primjer:

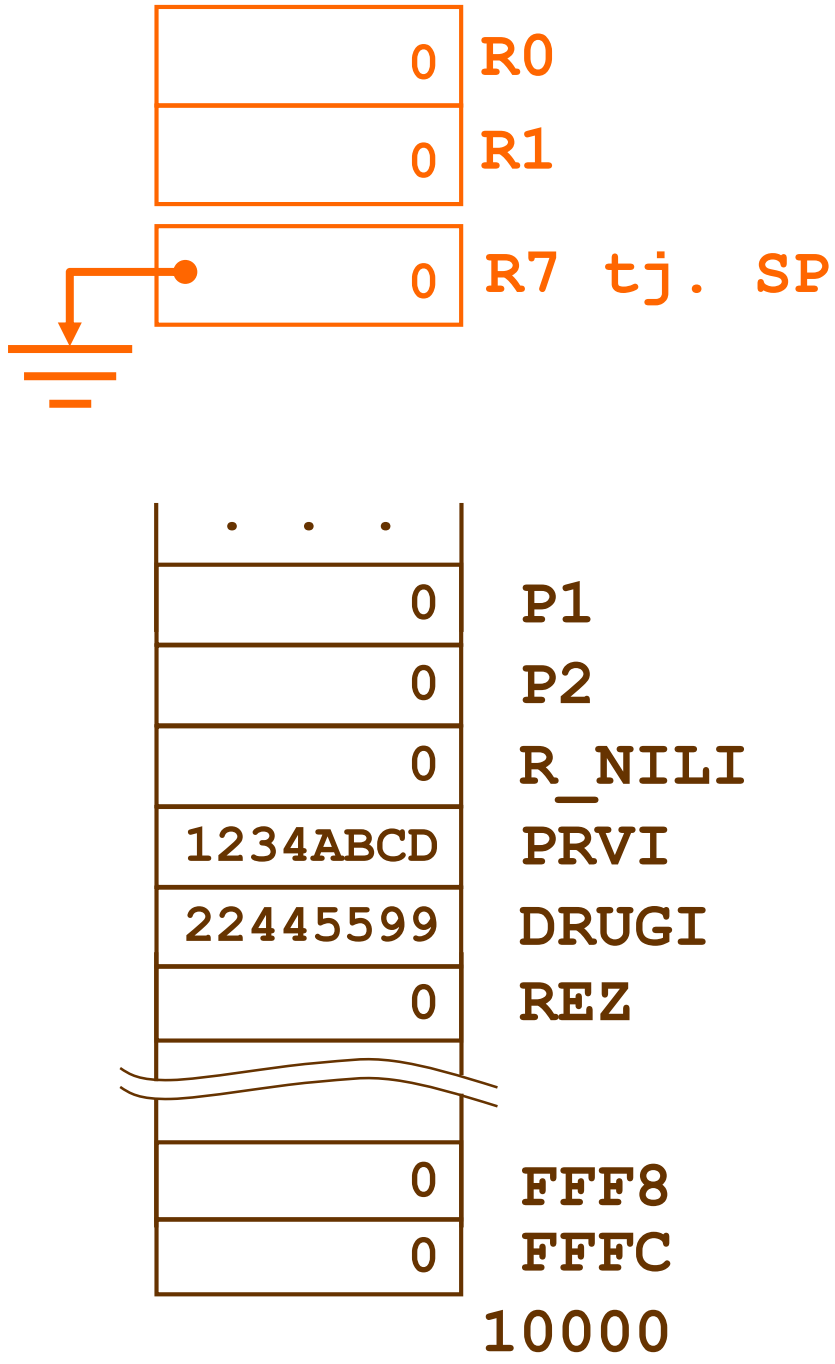
Napisati potprogram koji izračunava logičku operaciju NILI. Parametri se nalaze na memorijskim lokacijama P1 i P2, a rezultat se vraća lokacijom R_NILI. Glavni program iz memorije učitava dva podatka za koje računa NILI (pomoću potprograma) i rezultat sprema natrag u memoriju.

Rješenje:

`na sljedećem slajdu >>>>`

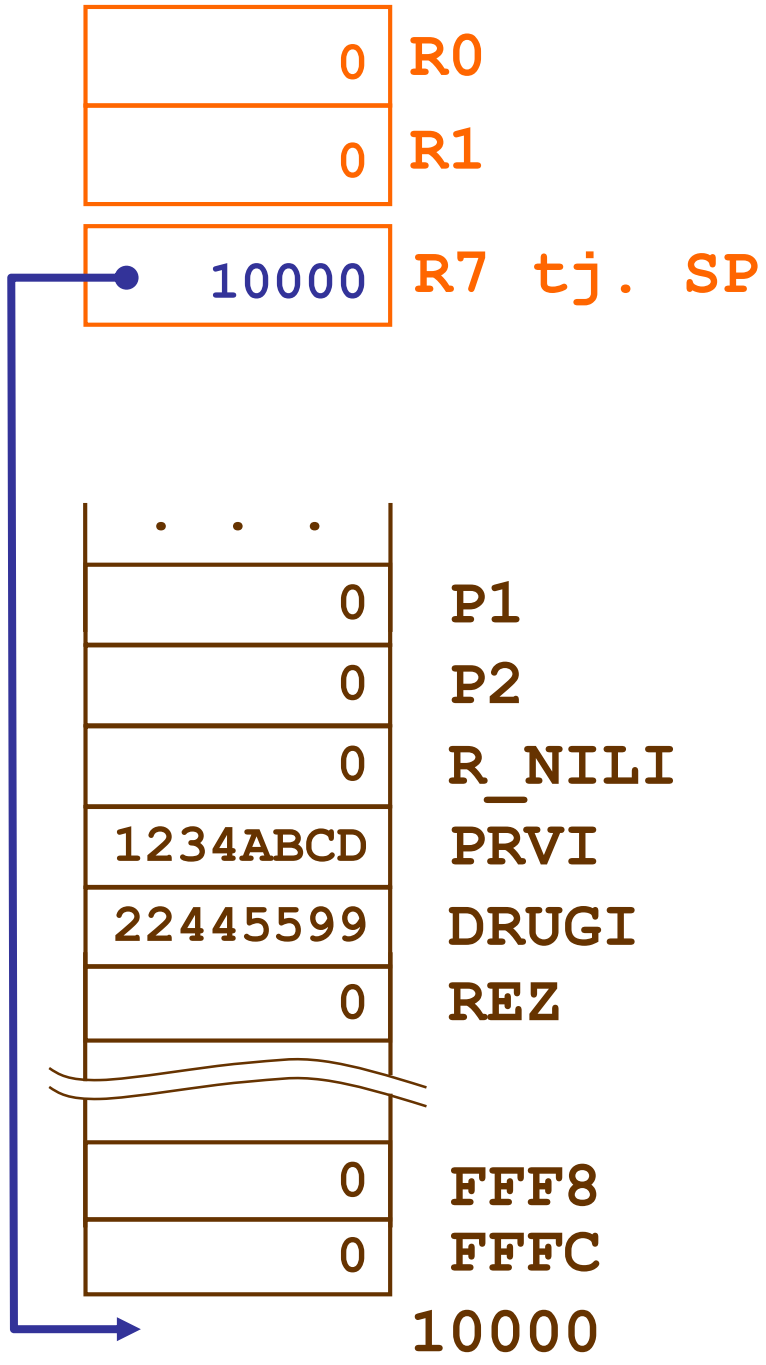
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



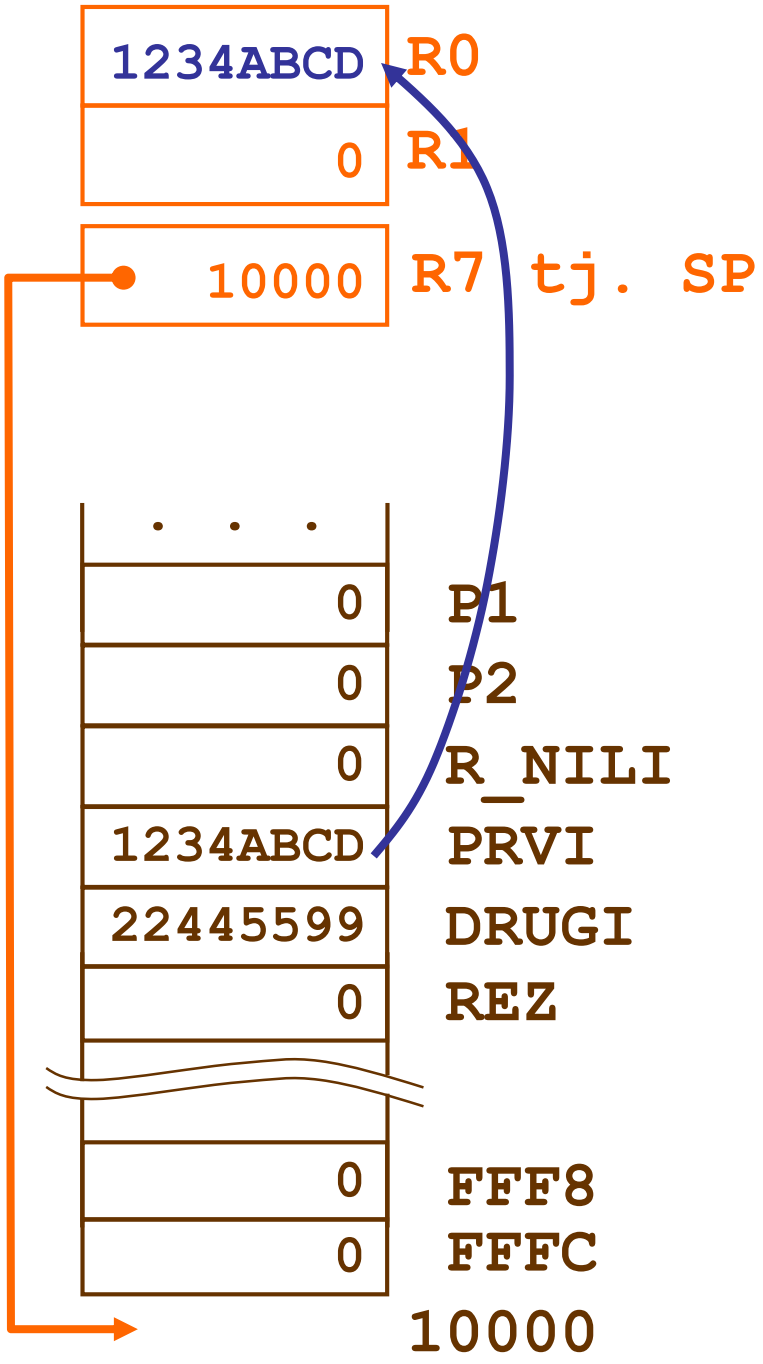
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



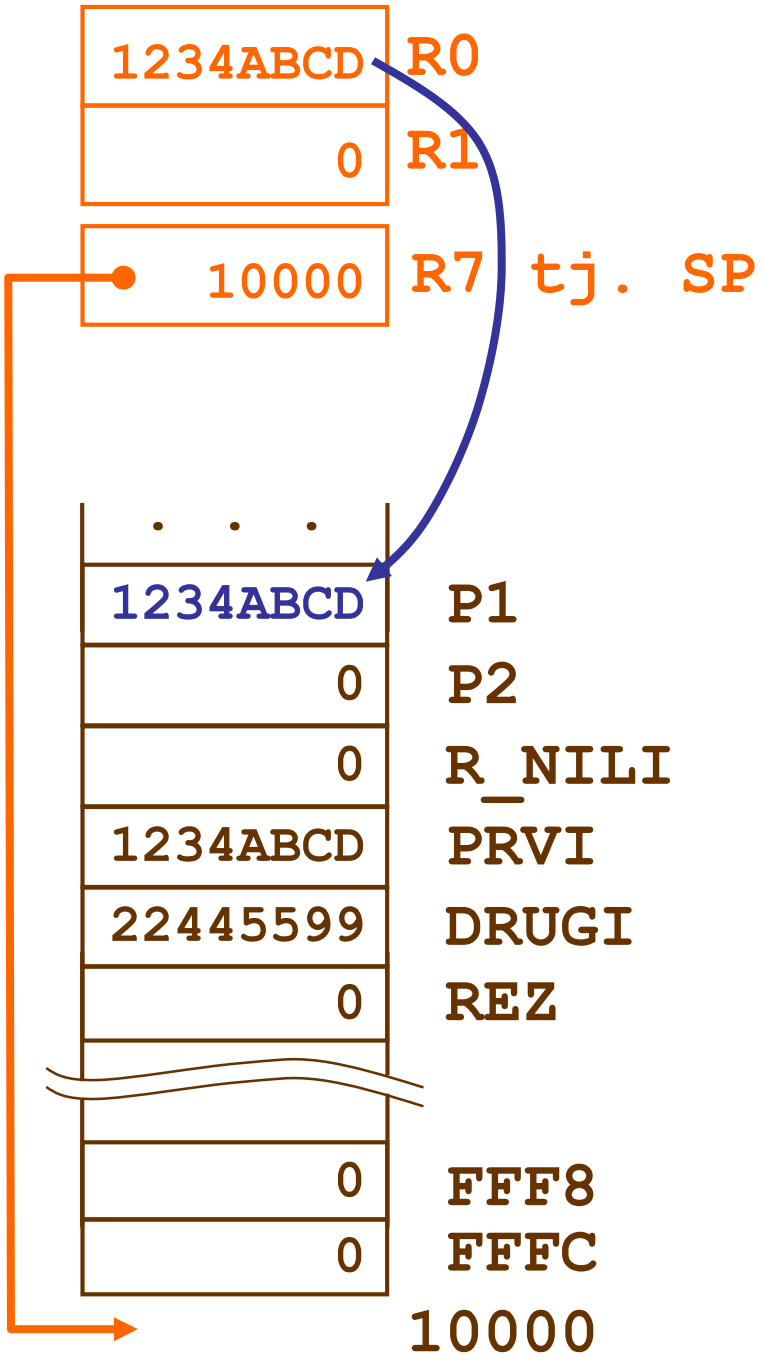
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



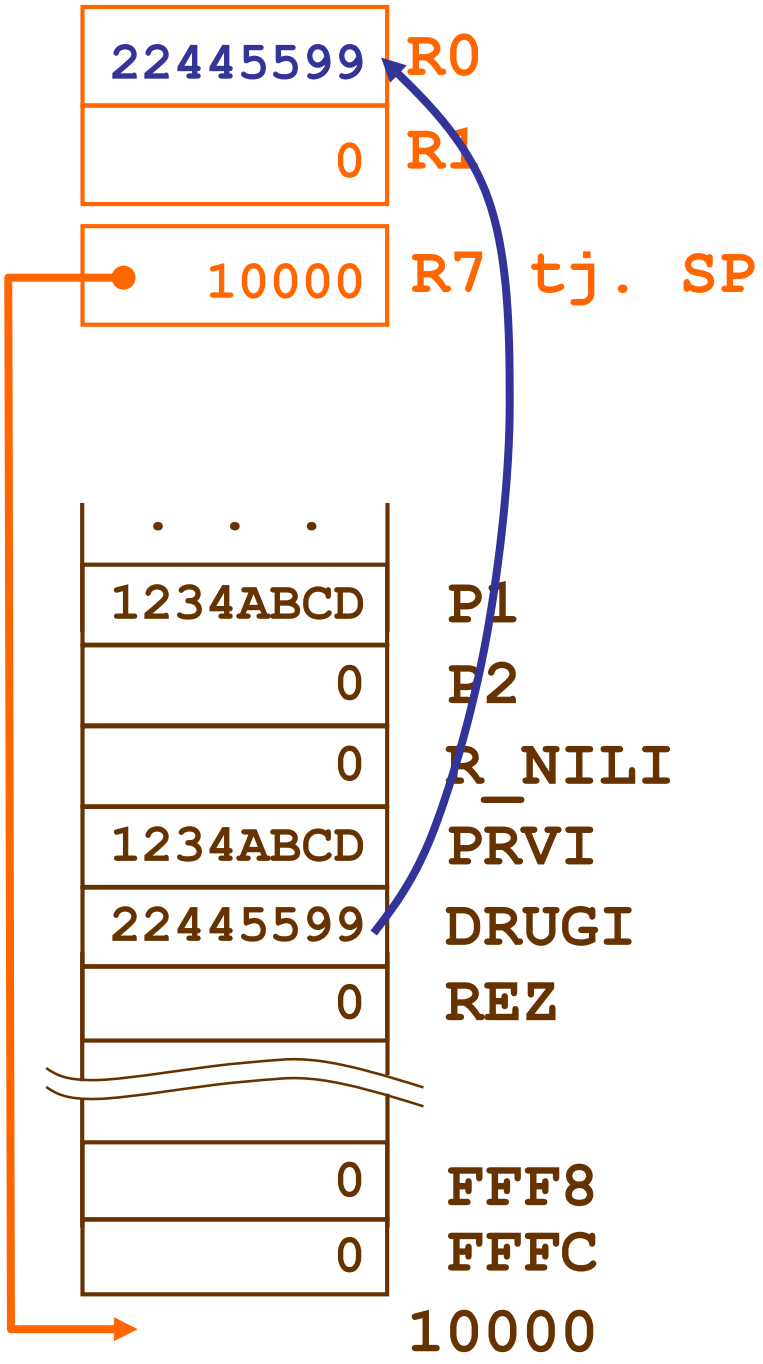
<<<< Izvođenje programa:

```
GLAVNI MOVE   10000, SP

       LOAD   R0, (PRVI)
       STORE  R0, (P1)

       LOAD   R0, (DRUGI)
       STORE  R0, (P2)

       CALL   NILI

       LOAD   R0, (R_NILI)
       STORE  R0, (REZ)

       HALT
```



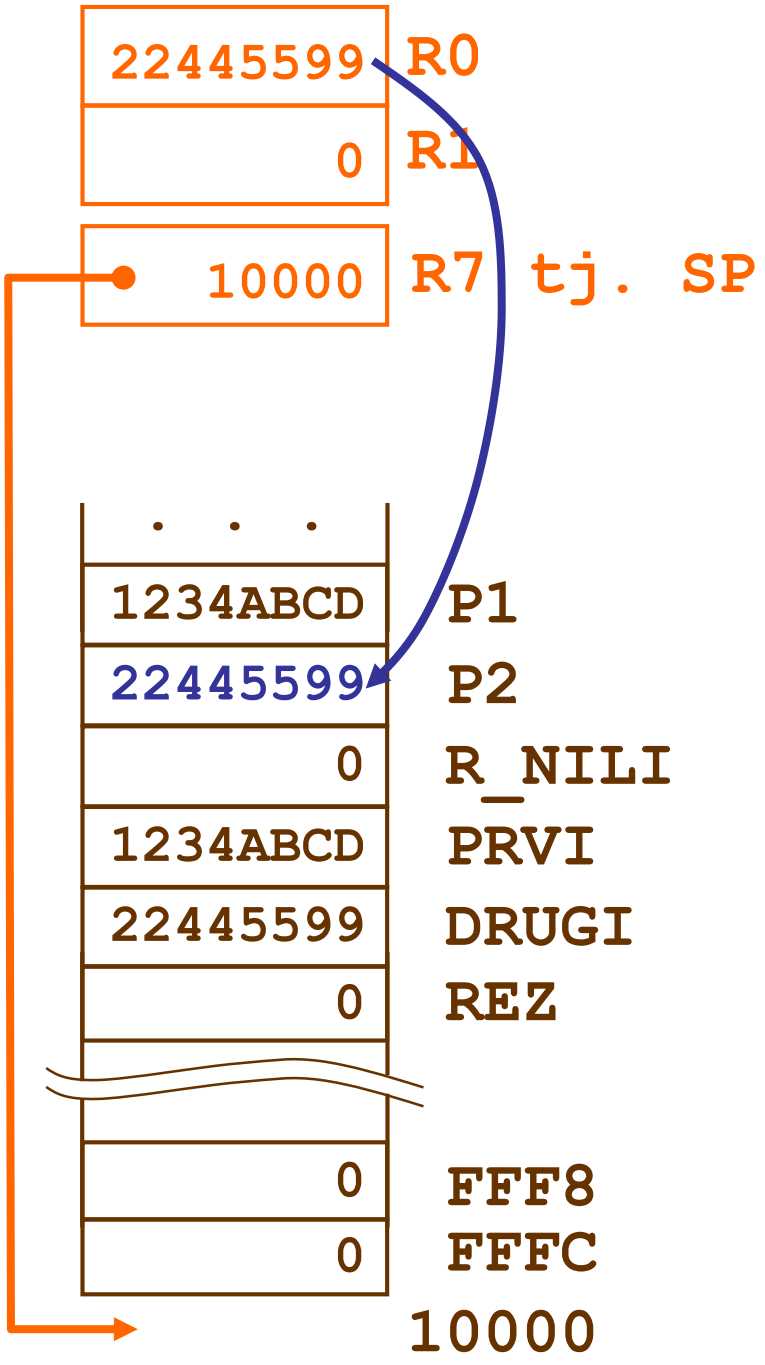
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



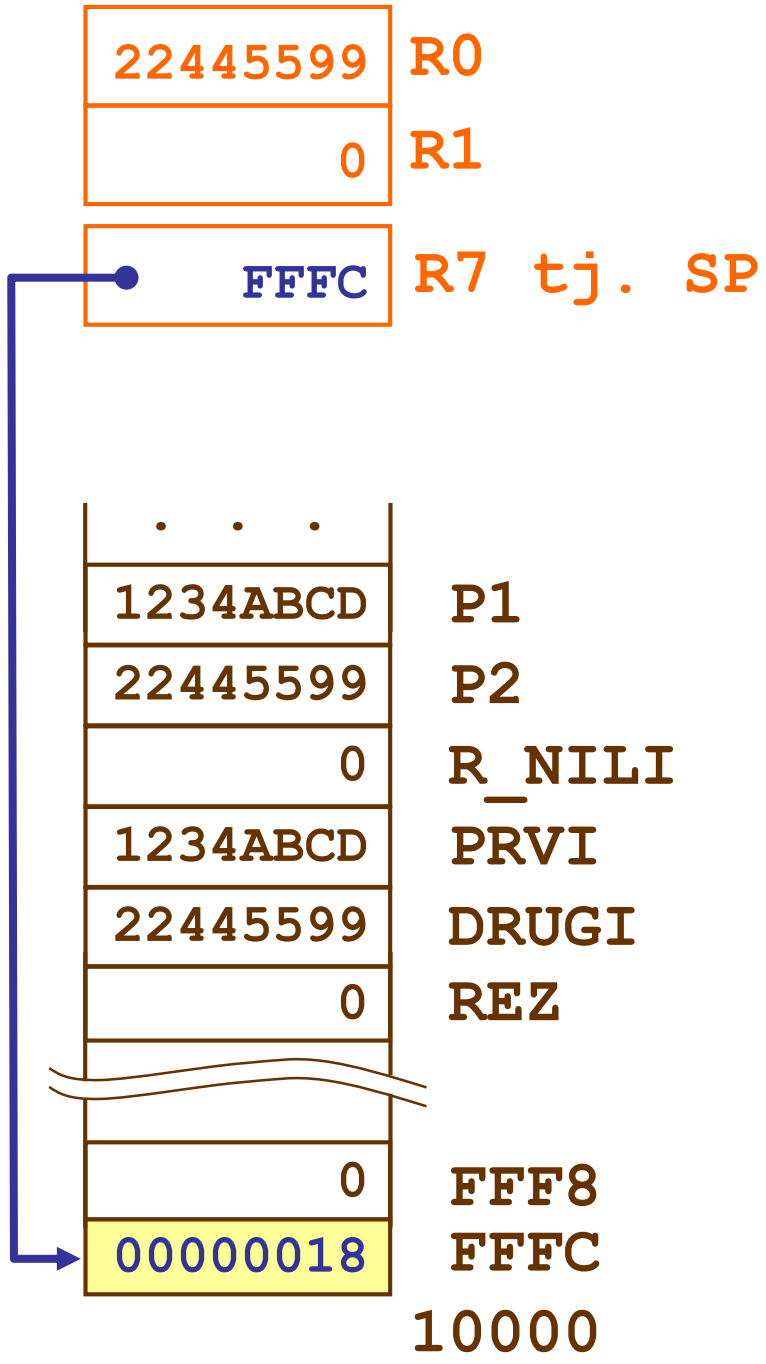
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



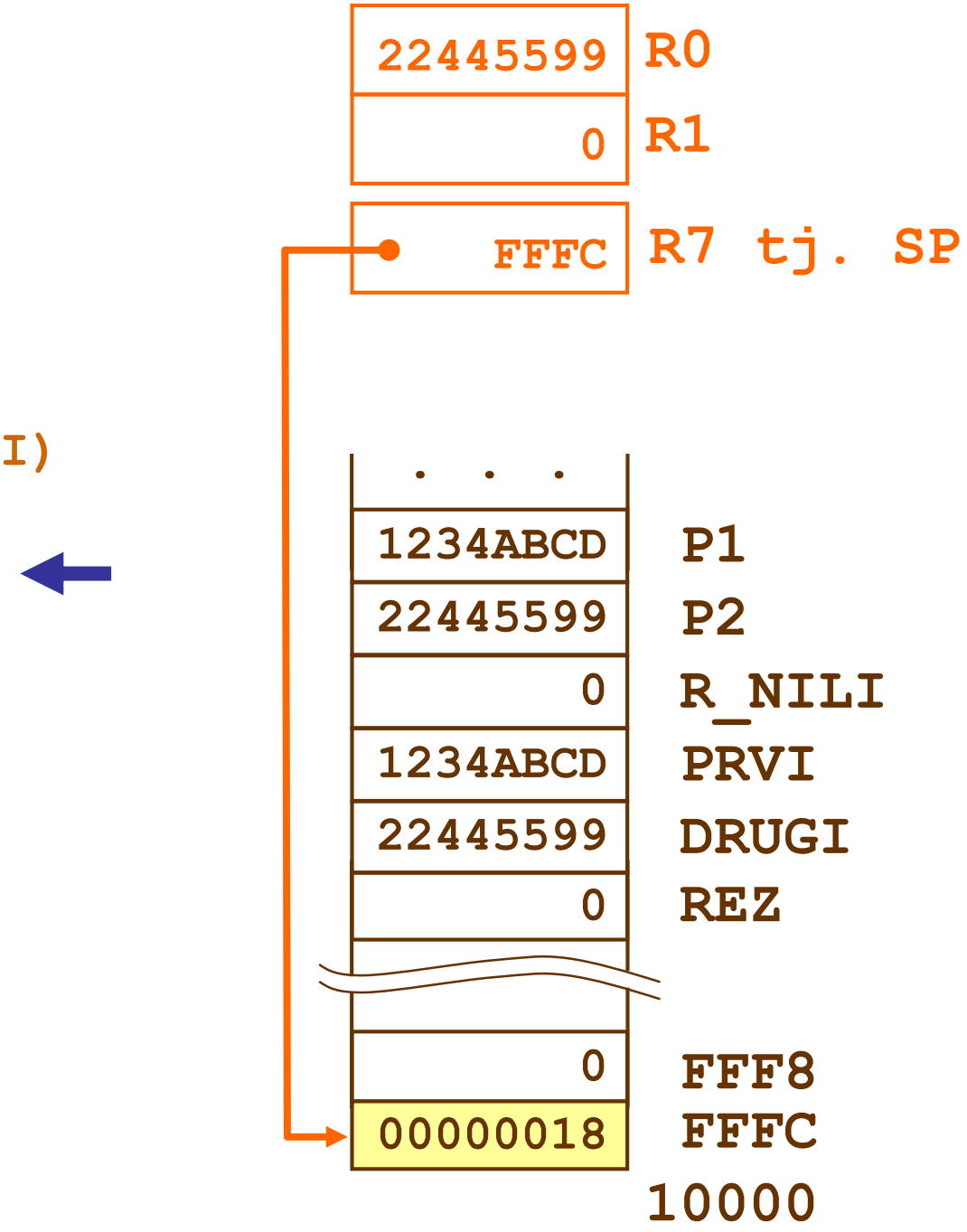
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



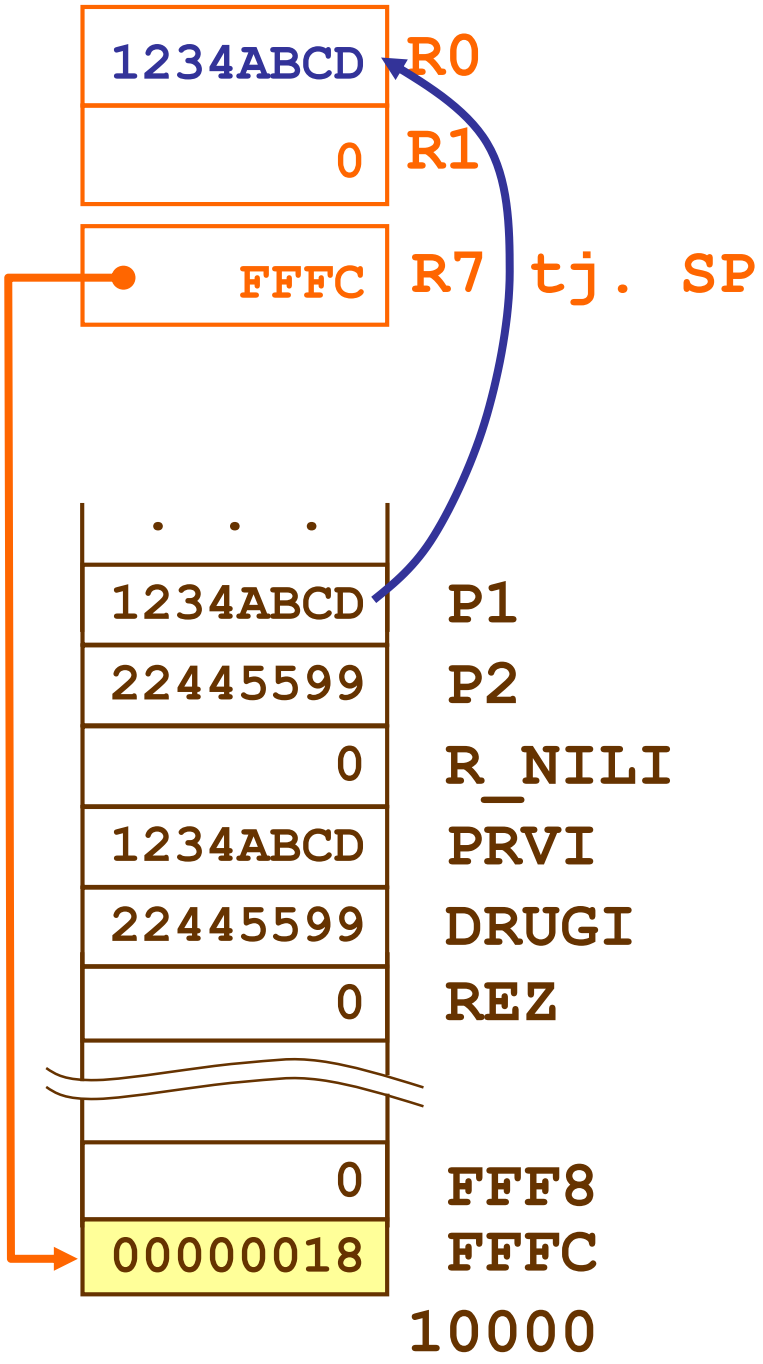
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)

        OR     R0, R1, R0
        XOR    R0, -1, R0

        STORE  R0, (R_NILI)
        RET
```



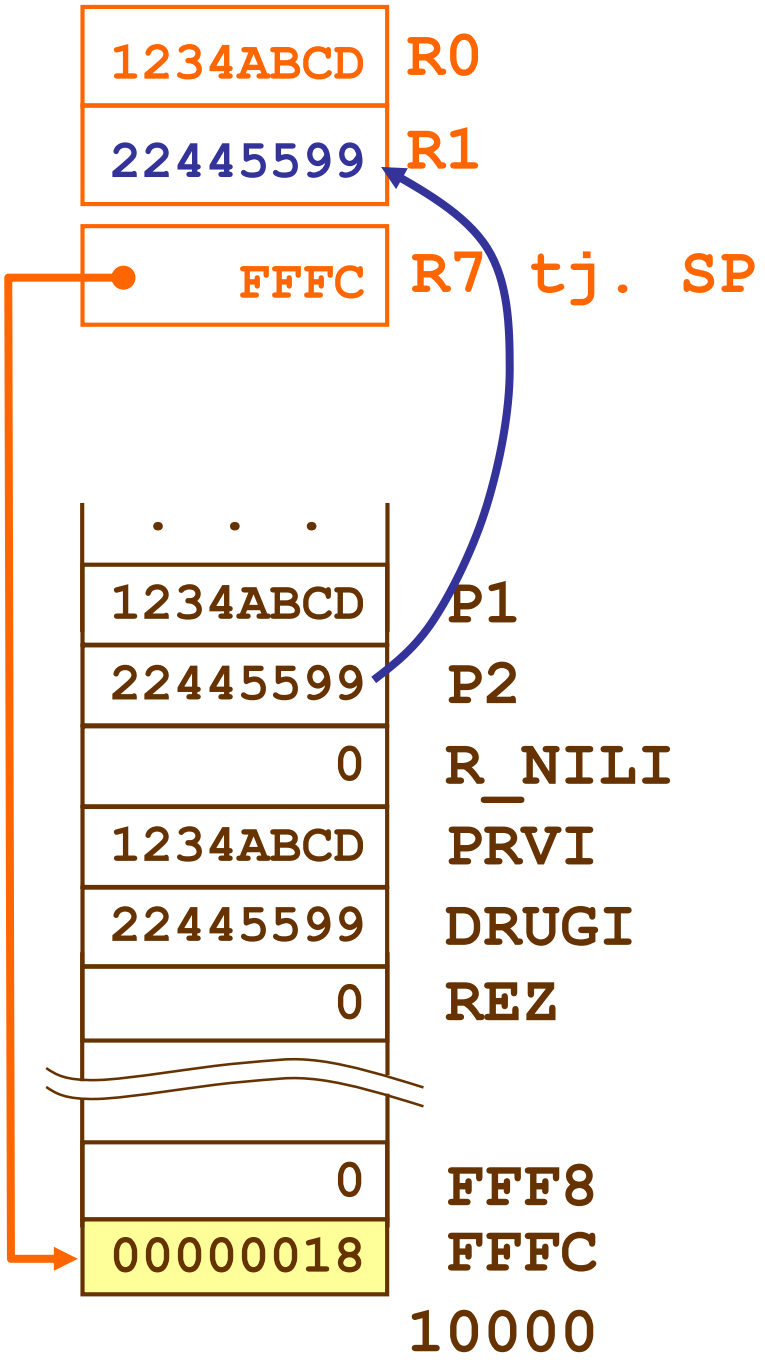
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)

        OR     R0, R1, R0
        XOR    R0, -1, R0

        STORE  R0, (R_NILI)
        RET
```



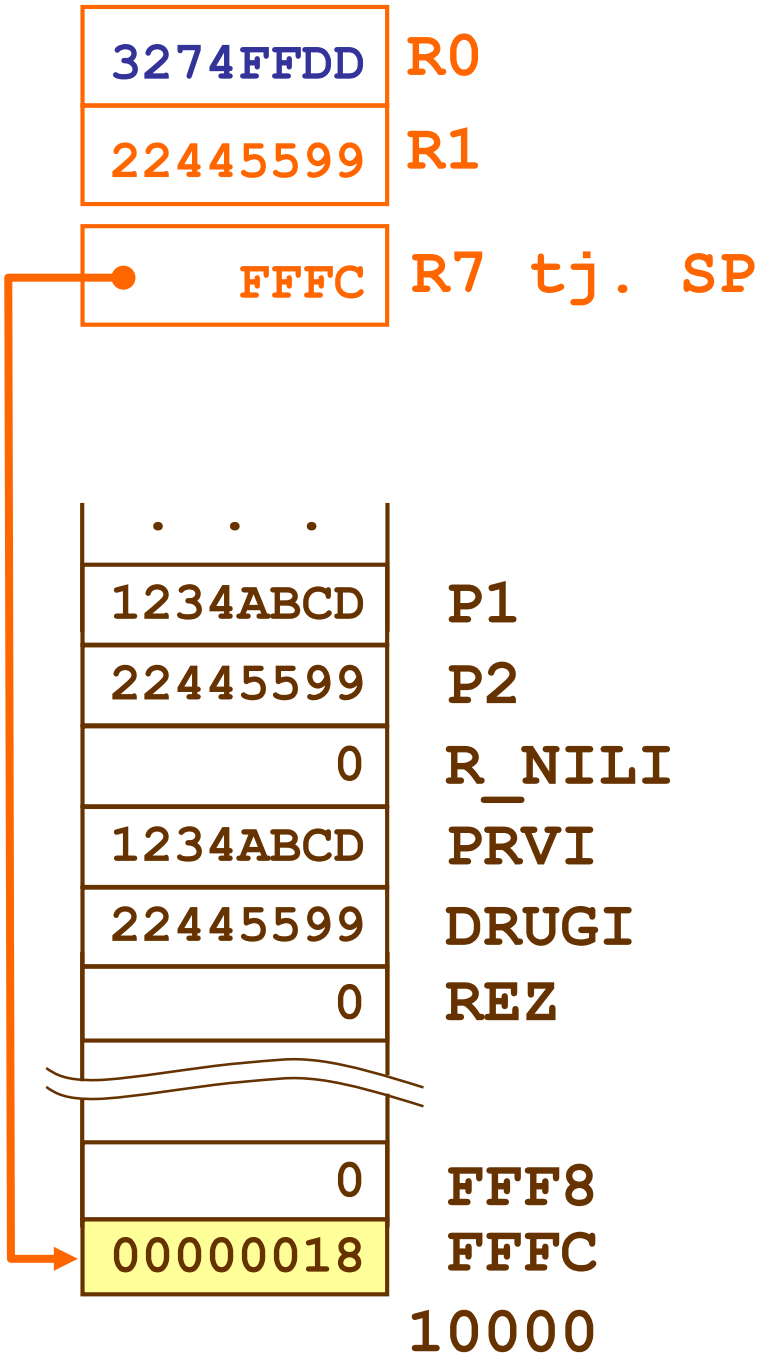
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)
        OR     R0, R1, R0
        XOR    R0, -1, R0
        STORE  R0, (R_NILI)
        RET
```



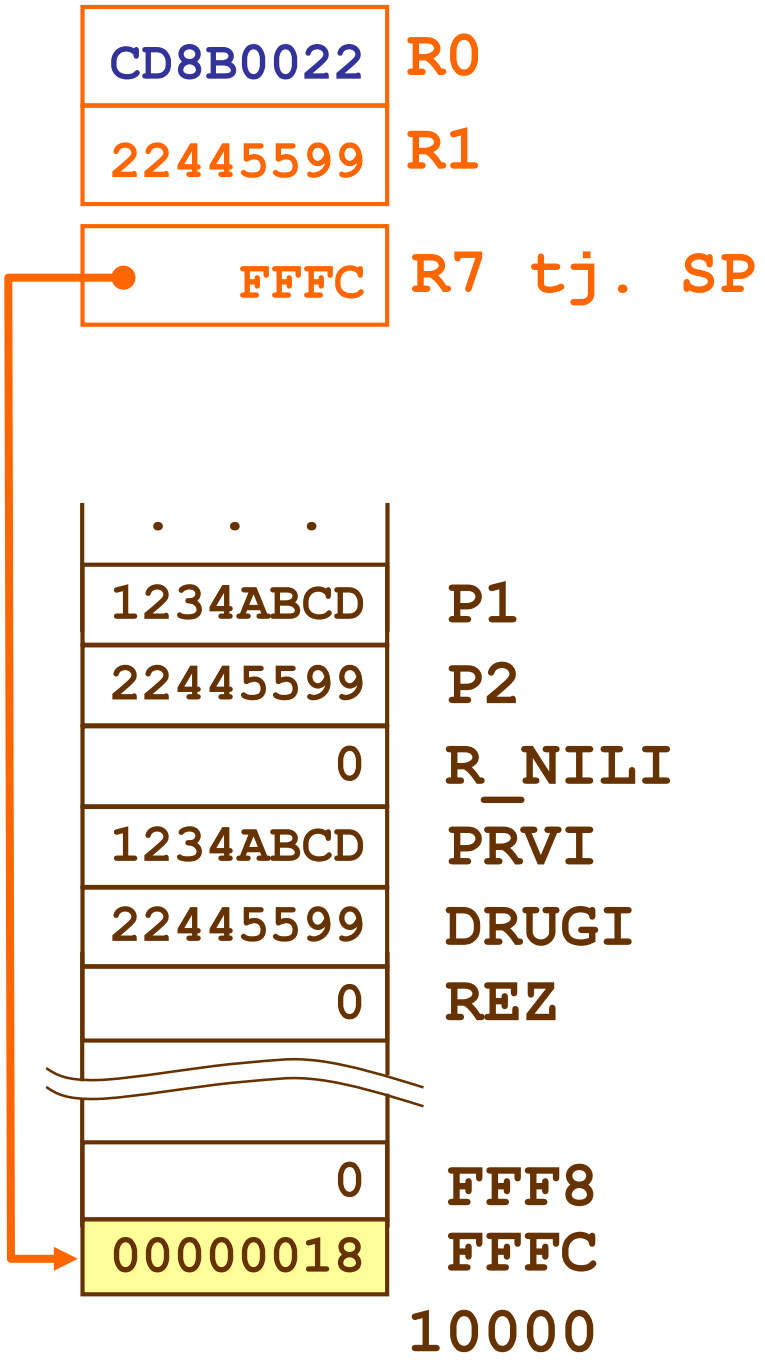
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)

        OR     R0, R1, R0
        XOR    R0, -1, R0

        STORE  R0, (R_NILI)
        RET
```



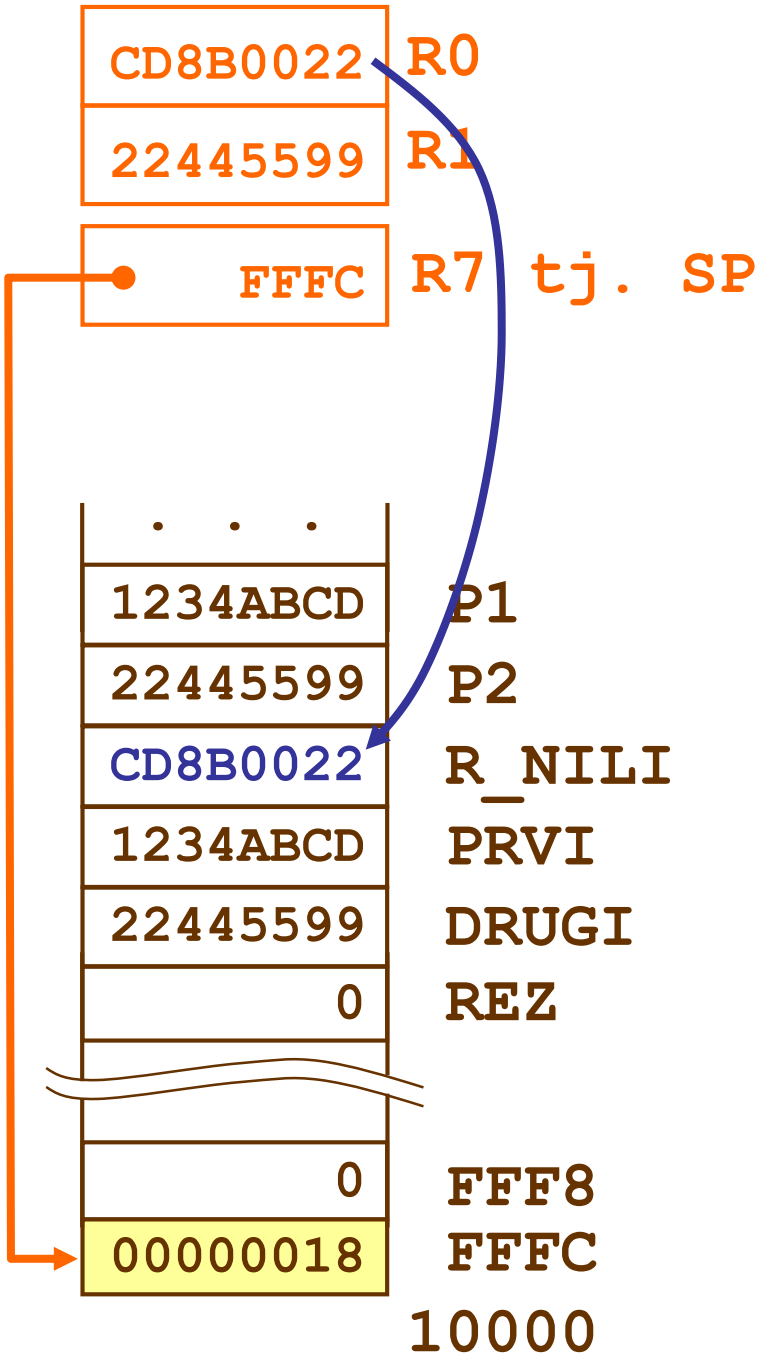
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)

        OR     R0, R1, R0
        XOR    R0, -1, R0

        STORE  R0, (R_NILI)
        RET
```



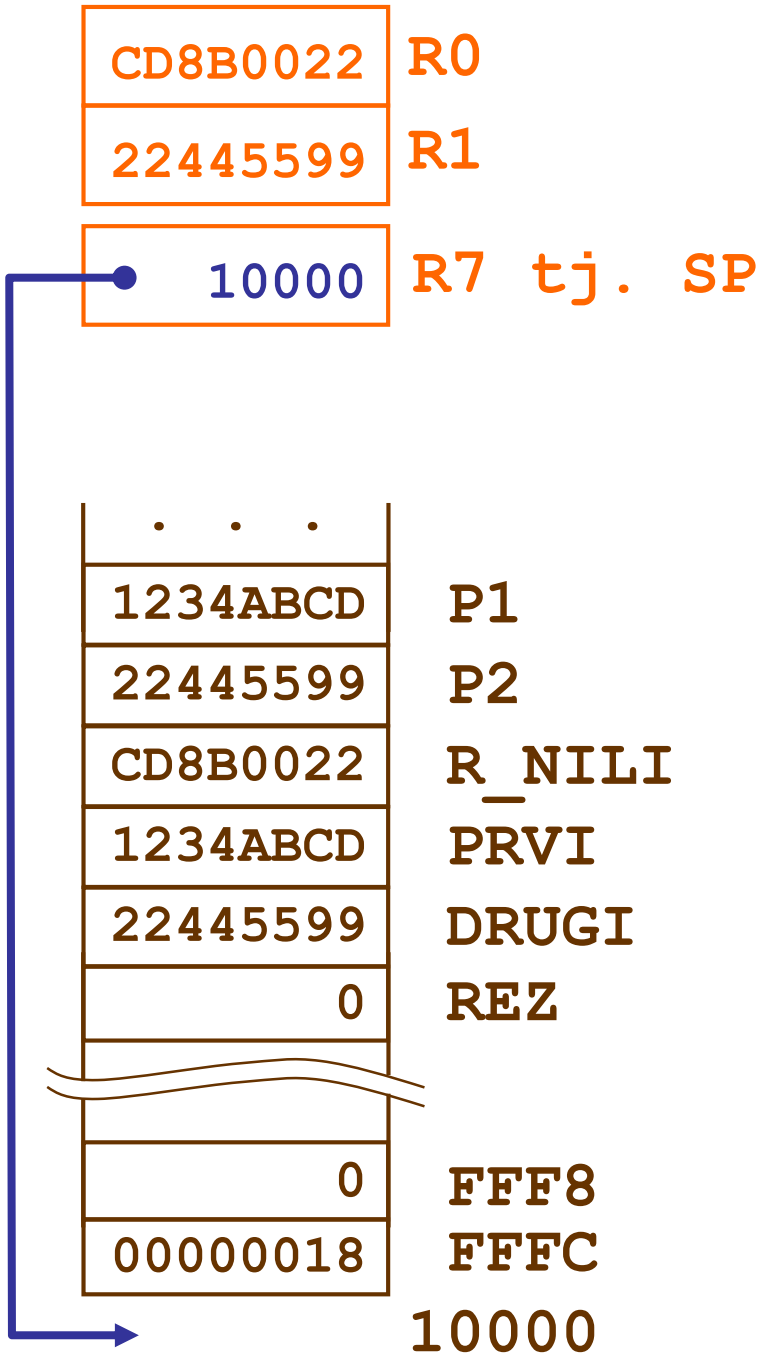
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)

        OR     R0, R1, R0
        XOR    R0, -1, R0

        STORE  R0, (R_NILI)
        RET
```



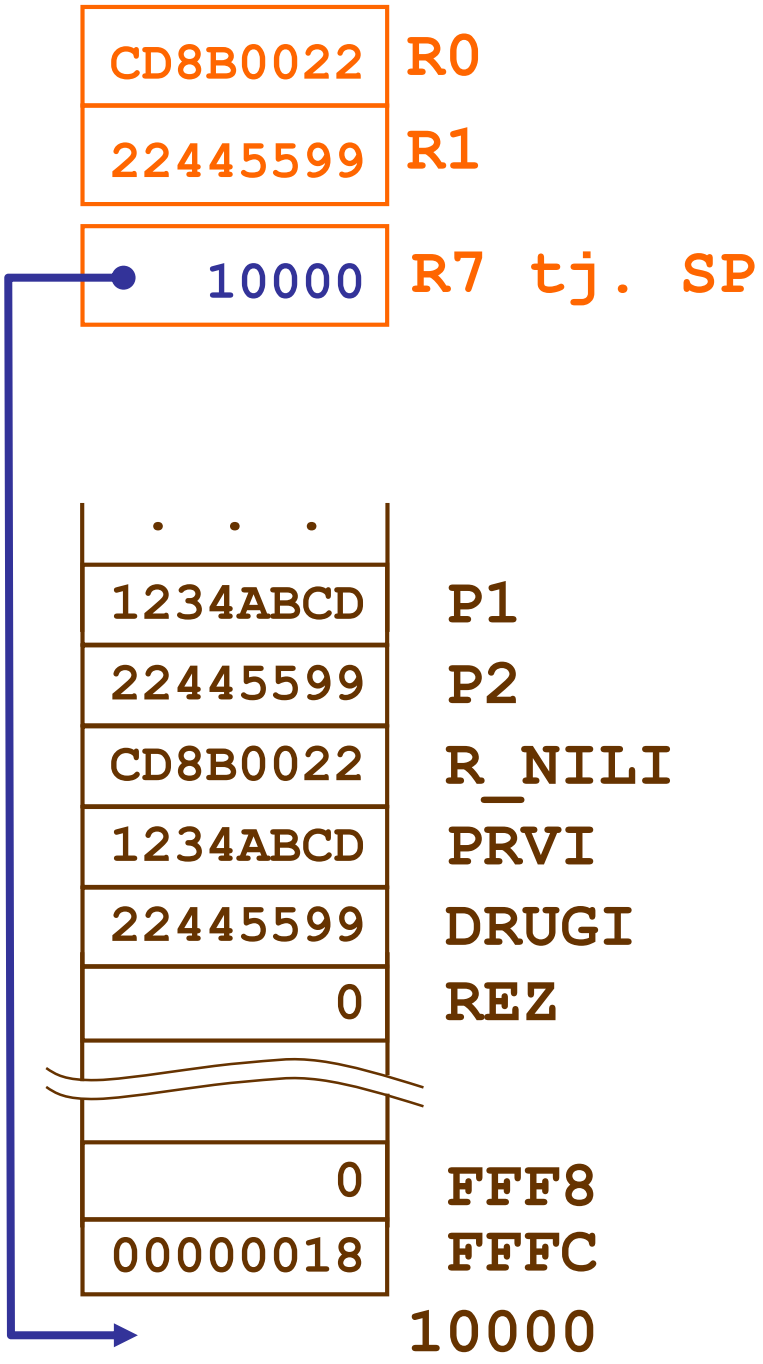
<<<< Izvođenje programa:

```
NILI    LOAD   R0, (P1)
        LOAD   R1, (P2)

        OR     R0, R1, R0
        XOR    R0, -1, R0

        STORE  R0, (R_NILI)
        RET
```



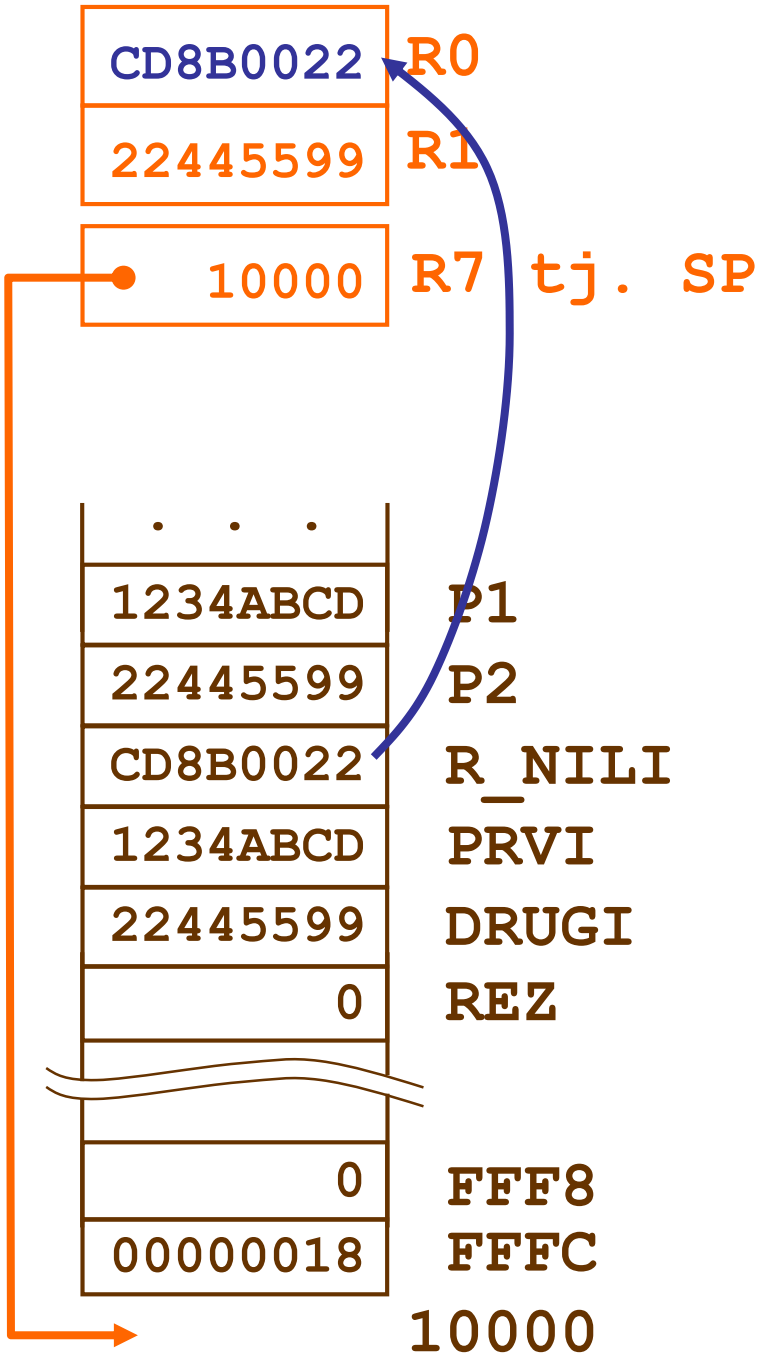
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



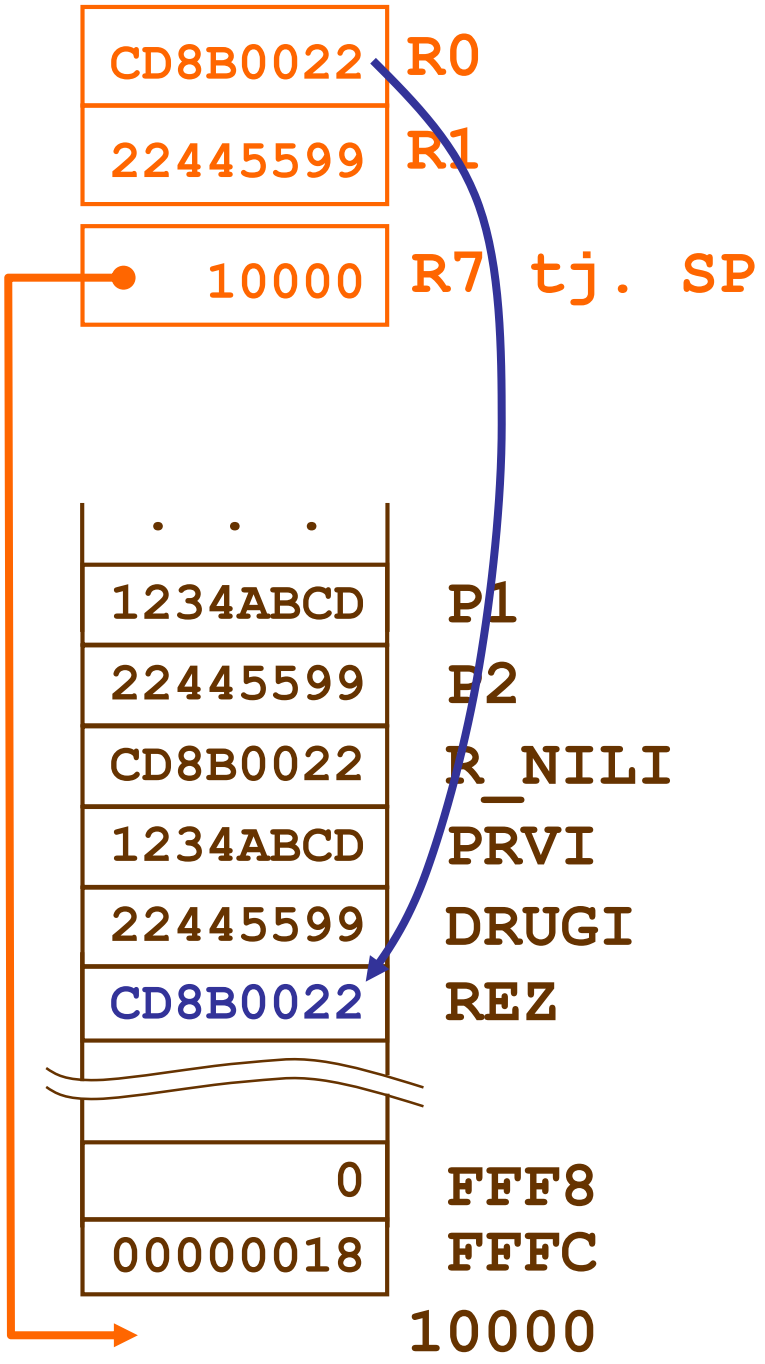
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



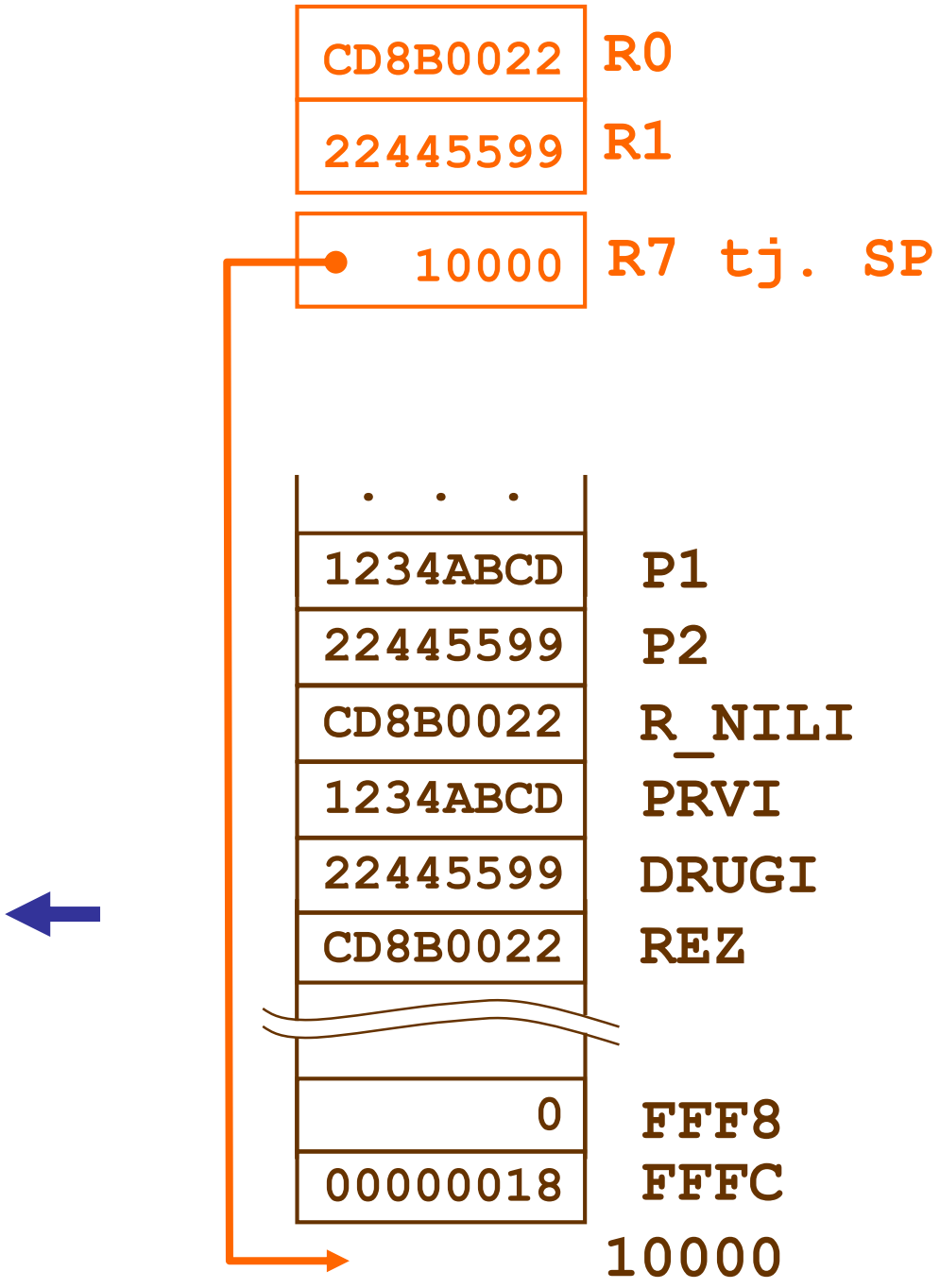
<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



<<<< Izvođenje programa:

```
GLAVNI MOVE  10000, SP

       LOAD  R0, (PRVI)
       STORE R0, (P1)

       LOAD  R0, (DRUGI)
       STORE R0, (P2)

       CALL  NILI

       LOAD  R0, (R_NILI)
       STORE R0, (REZ)

       HALT
```



<<<< (kompletan listing s komentarima)

```
        ; glavni program
GLAVNI  MOVE  10000, SP    ;važno: inicijaliziraj SP !!!

        ; stavi vrijednost u prvi parametar
        LOAD  R0, (PRVI)
        STORE R0, (P1)

        ; stavi vrijednost u drugi parametar
        LOAD  R0, (DRUGI)
        STORE R0, (P2)

        CALL  NILI          ;poziv potprograma

        ; uzmi rezultat i spremi ga
        LOAD  R0, (R_NILI)
        STORE R0, (REZ)     ;spremanje rezultata

        HALT
```

>>>>

```
<<<<
        ; potprogram NILI
NILI    LOAD  R0, (P1)      ; dohvat prvog parametra
        LOAD  R1, (P2)      ; dohvat drugog parametra

        OR    R0, R1, R0    ; Izračunavanje
        XOR   R0, -1, R0    ; rezultata.

        STORE R0, (R_NILI) ; upis rezultata u memoriju
        RET

        ; fiksne lokacije za parametre
        ; i povratnu vrijednost
P1      DW  0
P2      DW  0
R_NILI DW  0

        ; podatci i mjesto za rezultat
PRVI    DW  1234ABCD
DRUGI   DW  22445599
REZ     DW  0                              >>>>
```

Komentar

Prethodni potprogram ima značajan nedostatak, a to je promjena vrijednosti u registrima R0 i R1. Ukoliko je glavni program imao u njima podatke koji će mu još trebati, oni će nakon poziva potprograma biti izgubljeni i glavni program neće raditi ispravno.

Jedna (loša) mogućnost je da se za svaki potprogram zna koje registre mijenja. Tada se pri izradi programa mora voditi računa da na mjestima pozivanja potprograma u tim registrima ne budu nikakvi korisni podatci.

Bolje rješenje je da potprogram "sačuva" sadržaje registara koje će mijenjati.

Ovo je bio naš potprogram koji mijenja registre (R0 i R1):

```
NILI

        LOAD   R0, (P1)       ; dohvat prvog parametra
        LOAD   R1, (P2)       ; dohvat drugog parametra

        OR     R0, R1, R0     ; Izračunavanje
        XOR    R0, -1, R0     ; rezultata.

        STORE  R0, (R_NILI)   ; upis rezultata u memoriju



        RET
```

Ovo je potprogram koji sprema registre koje mijenja

Registri se spremaju na fiksne memorijske lokacije

```
NILI    STORE R0, (R0_SPREM)      ; spremi R0 i R1 na
        STORE R1, (R1_SPREM)      ; fiksne lokacije

        LOAD  R0, (P1)       ; dohvat prvog parametra
        LOAD  R1, (P2)       ; dohvat drugog parametra

        OR    R0, R1, R0     ; Izračunavanje
        XOR   R0, -1, R0     ; rezultata.

        STORE R0, (R_NILI)  ; upis rezultata u memoriju

        LOAD  R0, (R0_SPREM)      ; obnovi vrijednosti
        LOAD  R1, (R1_SPREM)      ; registara R0 i R1
        RET

R0_SPREM  DW  0          ; Dodatne lokacije za
R1_SPREM  DW  0          ; spremanje registara
```

Ovakvo spremanje onemogućuje rekurzivne pozive i zahtijeva definiranje posebnih memorijskih lokacija - nije idealno

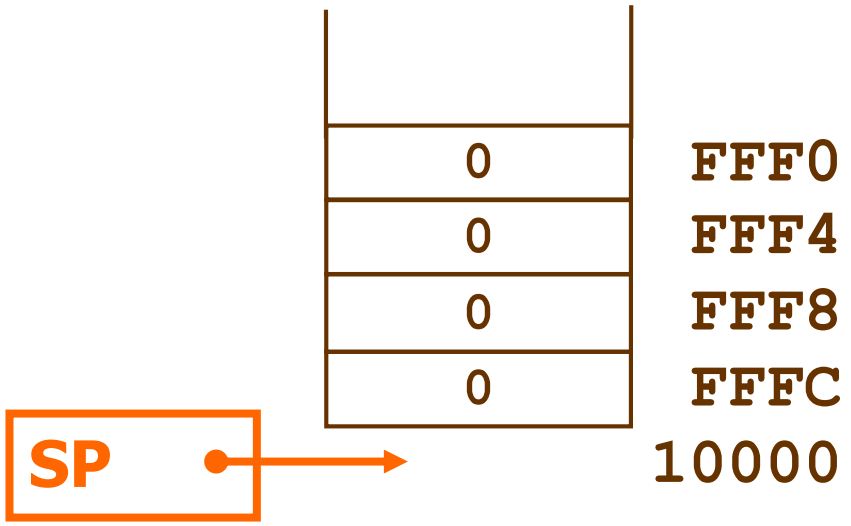
Bolje rješenje je spremanje na stog:

```
NILI    PUSH  R0       ; spremi R0 i R1
        PUSH  R1       ; na stog

        LOAD  R0, (P1)      ; dohvat prvog parametra
        LOAD  R1, (P2)      ; dohvat drugog parametra

        OR    R0, R1, R0    ; Izračunavanje
        XOR   R0, -1, R0    ; rezultata.

        STORE R0, (R_NILI) ; upis rezultata u memoriju

        POP   R1       ; obnovi vrijednosti registara
        POP   R0       ; R0 i R1 (OPREZ: REDOSLIJED!!!)
        RET
```

Ovakvo spremanje je bolje, omogućuje rekurzivne pozive i uobičajeno se koristi

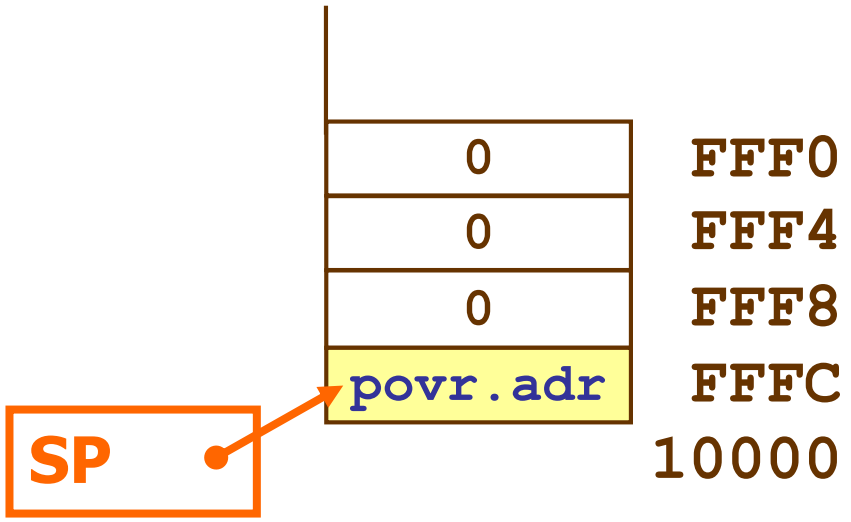
Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



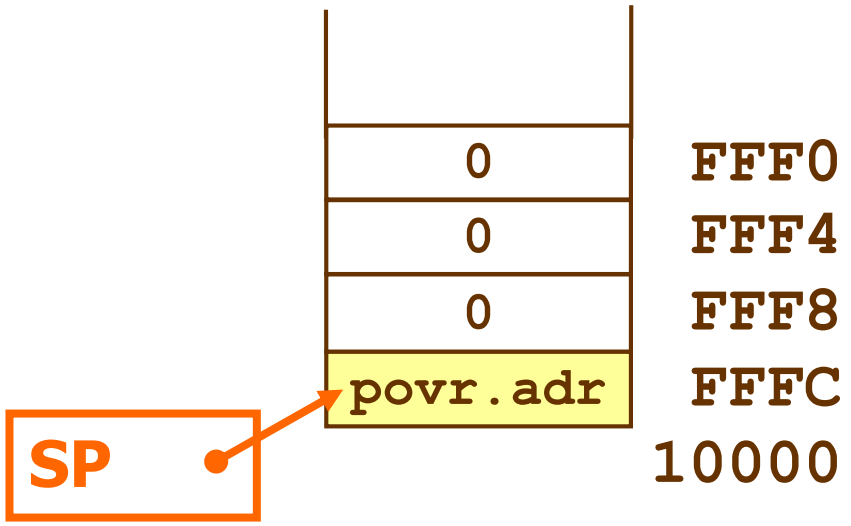
Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



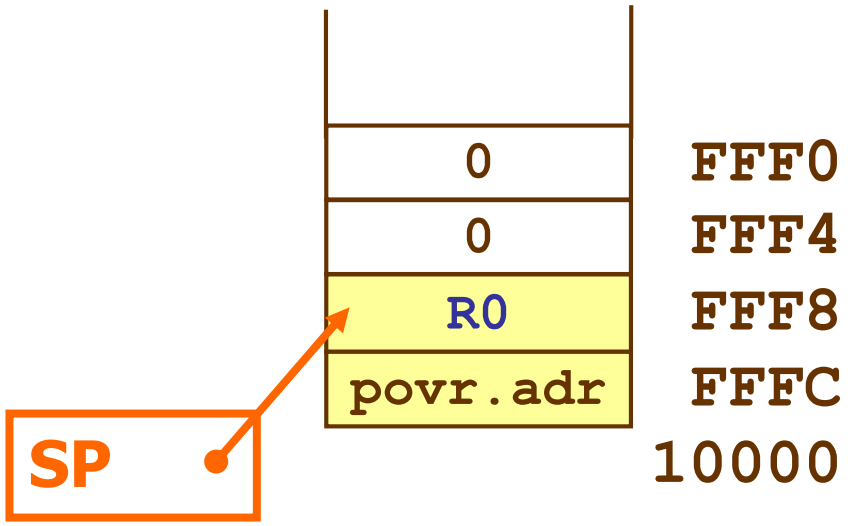
Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



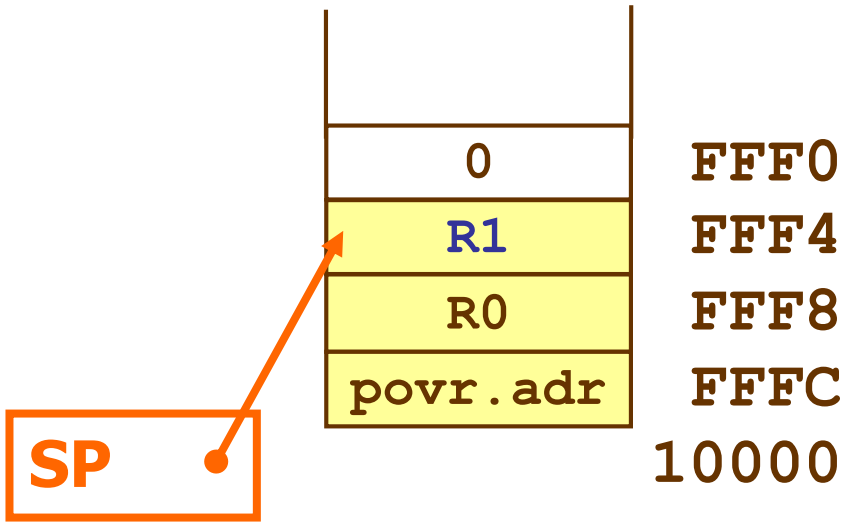
Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



Stog tijekom izvođenja:

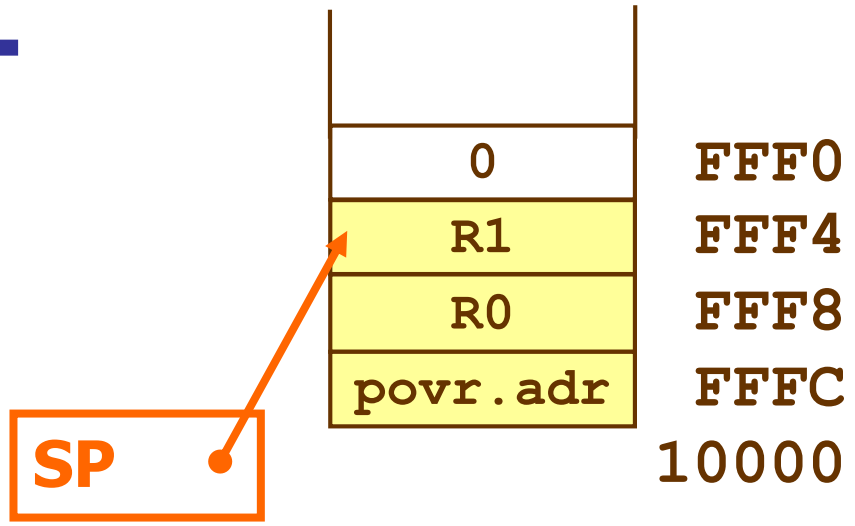
```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```

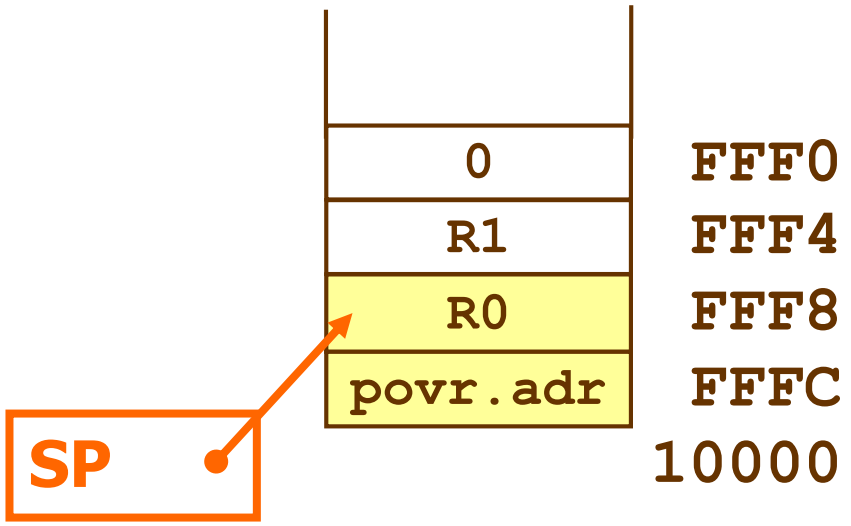
Registri R0 i R1 se mijenjaju. Stanje stoga se ne mijenja (smije se mijenjati, ako nakon naredaba potprograma stanje bude jednako kao prije njih)



Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```

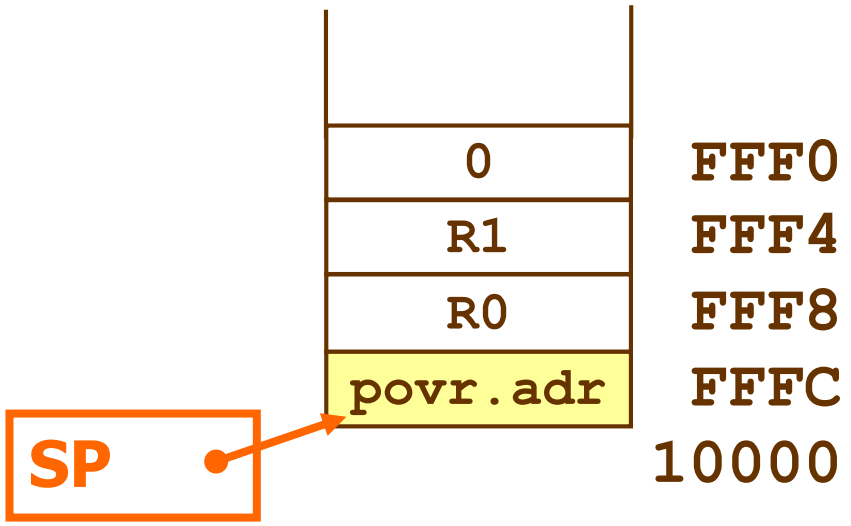
Registar R1 se obnavlja.



Stog tijekom izvođenja:

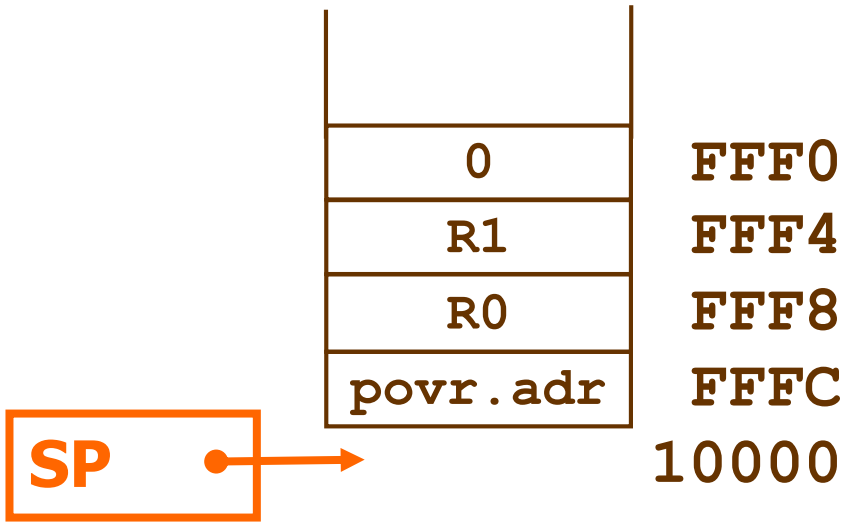
```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```

Registar R0 se obnavlja.



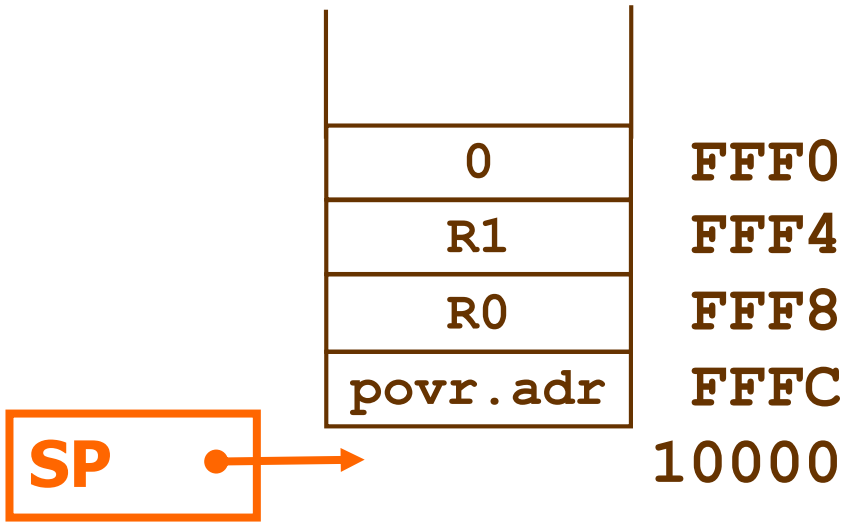
Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



Stog tijekom izvođenja:

```
GLAVNI  ...
        CALL   NILI
        LOAD   R0, (R_NILI)
        ...

NILI    PUSH   R0
        PUSH   R1

        ... ; naredbe
        ... ; potprograma

        POP    R1
        POP    R0
        RET
```



Potprogrami - Prijenos fiksnim lokacijama

- Prednosti:
  - Nema ograničenja na broj parametara (osim veličine memorije)
  - Registri su slobodni za druge namjene

- Nedostatci:
  - Sporiji rad s memorijskim lokacijama, nego s registrima (upis i čitanje)
  - Nešto duži i kompliciraniji program, nego kad se koriste registri
  - Nije moguće rekurzivno pozivanje potprograma