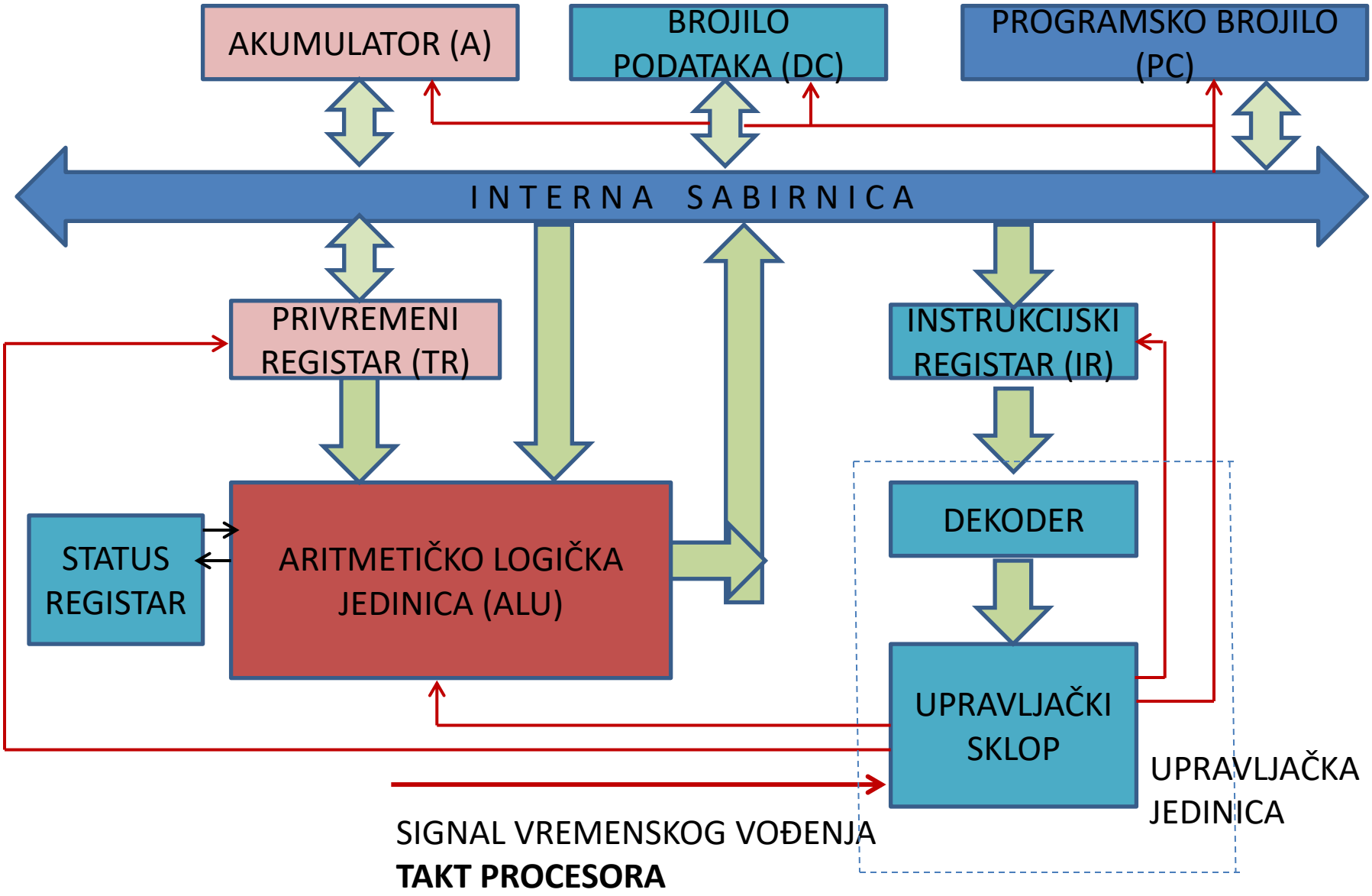
Građa računala

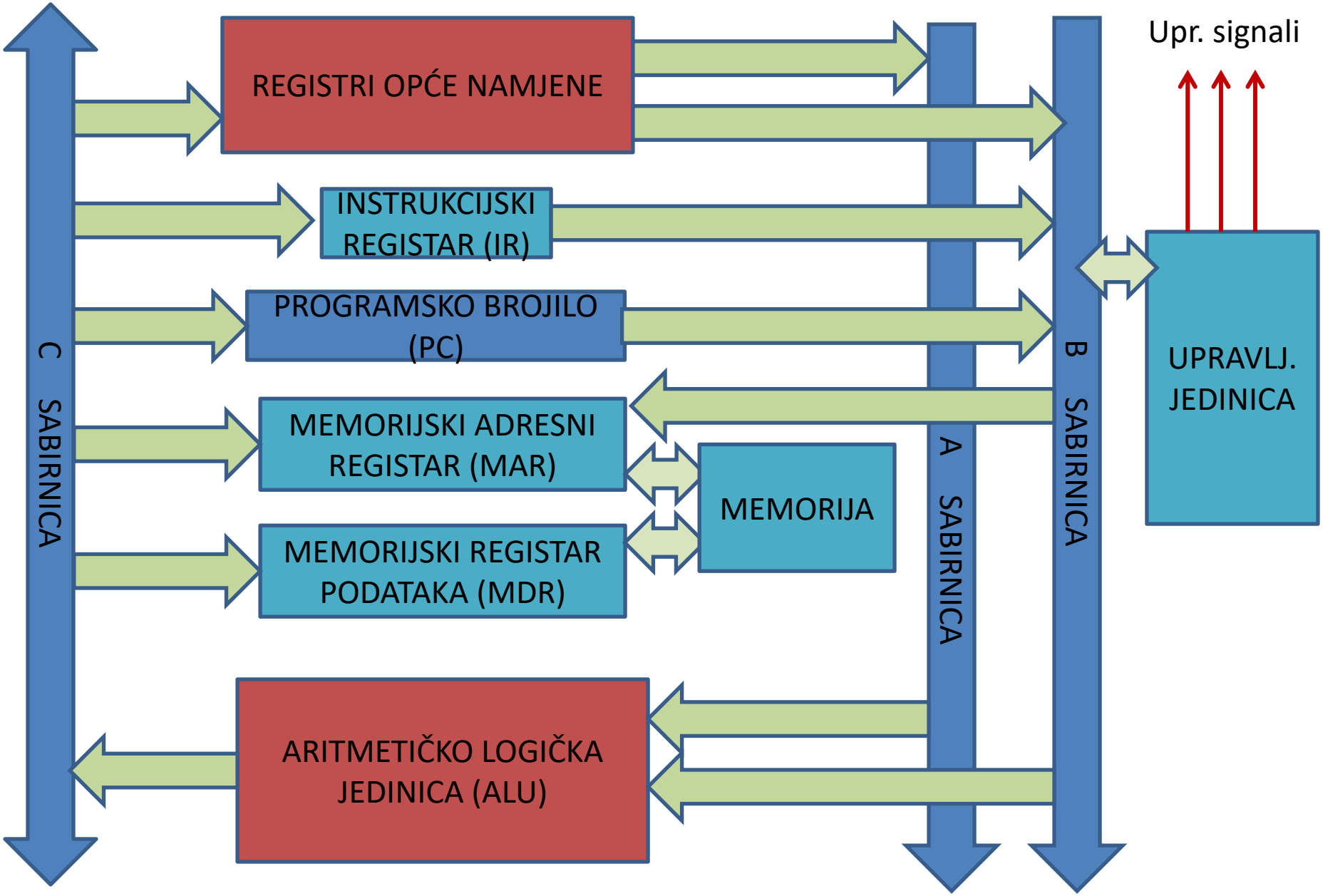
4. Upravljačka jedinica. Logičke funkcije

Preddiplomski izvanredni stručni studij
Informacijske tehnologije

Model procesora CISC arhitekture



Model procesora RISC arhitekture



Upravljačka jedinica

- generira upravljačke signale i koordinira sve aktivnosti unutar mikroprocesora
- sinkronizira:
  - komunikaciju između modula
  - prijenos podataka
- upravlja odgovorima na vanjske signale, npr.:
  - zahtjev za prekid
  - zaustavljanje procesora
  - stanje čekanja na odgovor memorije ili U/I uređaja

Osnovne funkcije upravljačke jedinice

- Uspostavljanje određenog stanja za vrijeme svakog strojnog ciklusa
- Ispravno određivanje sljedećeg stanja na temelju tekućeg stanja, stanja zastavica i stanja na ulaznim signalnim linijama mikroprocesora
- Pohranjivanje informacije koja opisuje tekuće stanje
- Na temelju stanja, generiranje upravljačkih signala za komunikaciju na modularnoj i međumodularnoj razini

Aktivnosti upravljačke jedinice za vrijeme faze PRIBAVI

- pribavlja instrukciju i dekodira operacijski kod
- mijenja stanje mikroprocesora i šalje upravljačke signale drugim komponentama mikroprocesora i mikroračunala
- slijedovi takvih upravljačkih signala upravljaju izvođenjem instrukcije pribavljene iz memorije
- element iz slijeda upravljačkih signala uzrok je jedne ili više osnovnih operacijana razini prijenosa podataka ili aktiviranja pojedinih sklopova

Osnovne vrste instrukcija kod RISC procesora

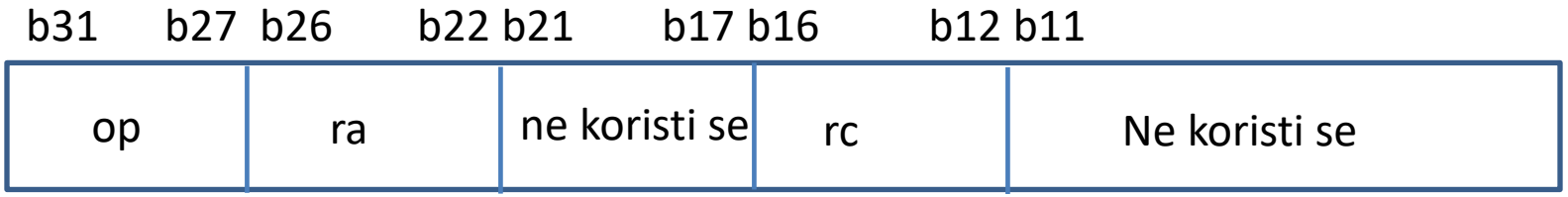
- aritmetičke
- logičke (logičke funkcije)
- naredbe prijenosa podataka (memorijske naredbe)
- naredbe grananja

Osnovne vrste instrukcija kod RISC procesora

- Ovisno o vrsti instrukcije, oblikuje se sadržaj instrukcijskog registra
- U nastavku se navodi nekoliko slučajeva za pojednostavljeni model 32-bitnog RISC procesora !

Aritmetičke ili logičke instrukcije

- Aritmetičke i logičke instrukcije možemo razvrstati na one koje specificiraju operacije s jednim operandom (***unarne***) i one koje određuju operacije s dva operanda (***binarne***)
- Format aritmetičke ili logičke instrukcije s jednim operandom prikazan je na slici:

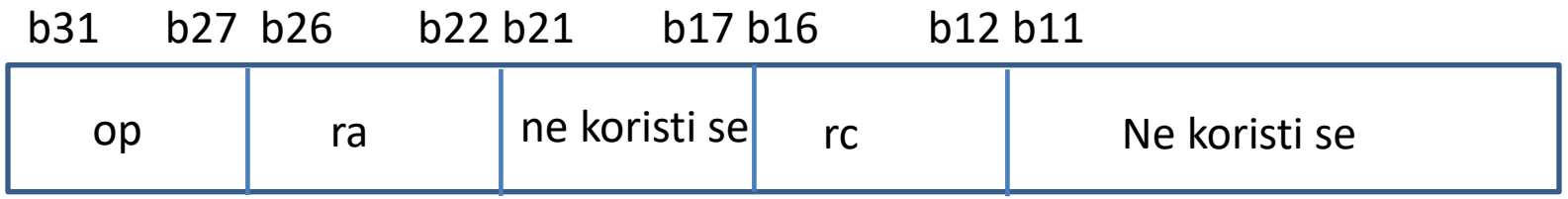


Aritmetičke ili logičke instrukcije

- **Primjeri aritmetičkih ili logičkih funkcija s jednim operandom:**
  - Aritmetička instrukcija kojom se izvodi potpuni komplement na operandu dohvaćenom iz registra R[rc] i rezultat smješta u registar R[ra]

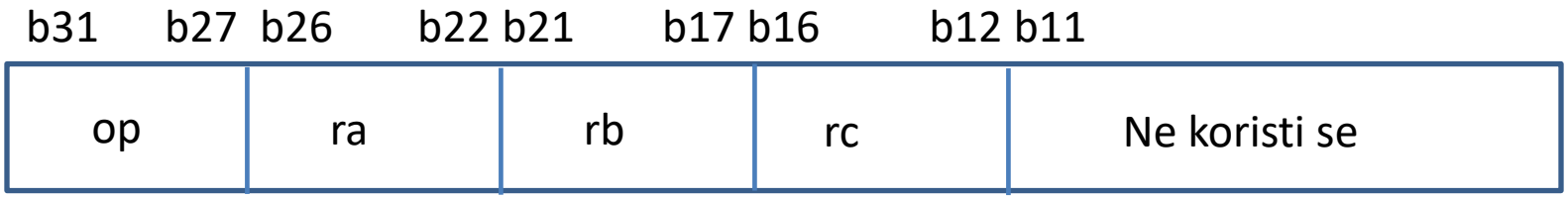
    neg ra, rc *; R[ra] <- R[rc]*
  - Logička instrukcija koja iz registra R[rc] dohvaća operand, pretvara ga u jedinični komplement i smješta ga u odredišni registar R[ra]

    not ra, rc *; R[ra] <- R[rc]*



Aritmetičke ili logičke instrukcije

- Format aritmetičke ili logičke instrukcije s dva operanda prikazan je na slici:



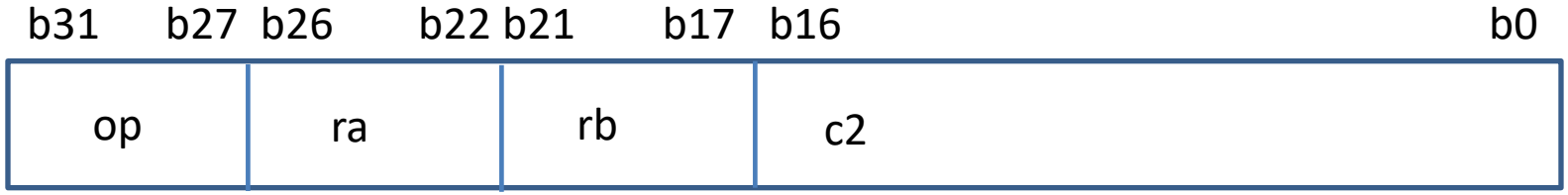
- Grupu aritmetičkih i logičkih instrukcija s dvama operandima čine instrukcije:
  - ***add ra, rb, rc*** *; R[ra] <- R[rb] + R[rc]*
  - ***sub ra, rb, rc*** *; R[ra] <- R[rb] - R[rc]*
  - ***and ra, rb, rc*** *; R[ra] <- R[rb] AND R[rc]; pri čemu AND označava logičko I*
  - ***or ra, rb, rc*** *; R[ra]<- R[rb] OR R[rc]; pri čemu OR označava logičko ILI*

Instrukcije za pristup memoriji

- Arhitektura RISC vrlo se često naziva i ***load-store*** arhitektura
- Jedna od važnih značajki RISC procesora da se jedino strojnim instrukcijama vrste load i store može pristupiti memoriji
- Model RISC procesora ima četiri instrukcije *load*, od toga dvije "prave" (*ld* i *ldr*) - za dohvat operanda iz memorije i smještanje u jedan od registara u skupu registara opće namjene

Instrukcije za pristup memoriji

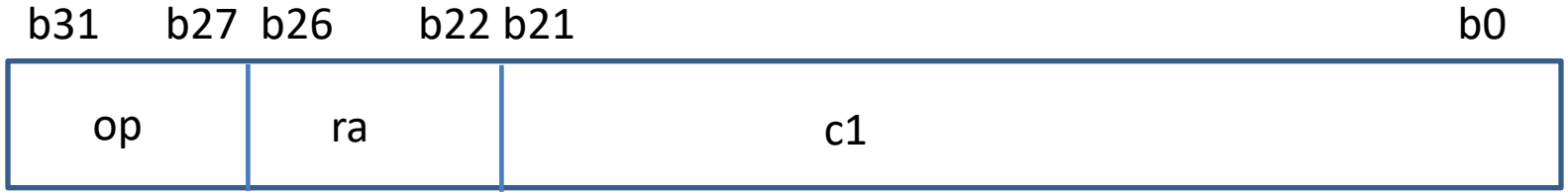
- Format instrukcije *ld* (**load**) za pristup memoriji prikazan je na slici:



- Instrukcija se izvodi tako da se dohvaćeni podatak iz memorije pohranjuje u odredišni registar koji je određen 5-bitnim poljem *ra*
- Adresa memorijske lokacije s koje se dohvaća podatak određuje se na temelju 17-bitne vrijednost koja je smještena u polju *c2*

Instrukcije za pristup memoriji

- Format instrukcije *ldr* (**load relative**) za pristup memoriji prikazan je na slici:



- Ima oblik:

  ldr ra, cl ;R[ra] <- M[PC + c1]

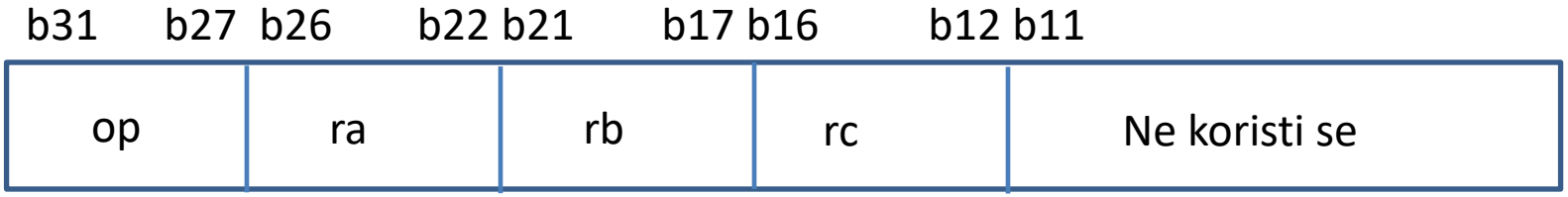
- Efektivna se adresa izvorišta operanda PC + c1 računa tijekom izvođenja programa i relativna je u odnosu na trenutni sadržaj programskog brojila PC

Primjer aritmetičke instrukcije

- **Zadatak:** Za zadani format aritmetičke instrukcije s dvama operandima pojednostavljenog modela 32-bitnog RISC procesora odredite vrijednosti pojedinih bitovnih polja za aritmetičku instrukciju

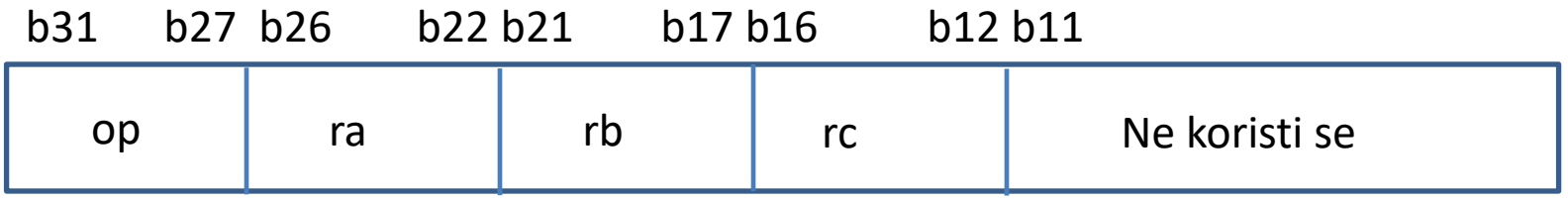
  add r1, r2, r28

- Operacijski kod instrukcije *add* je $(01100)_2$



Primjer aritmetičke instrukcije

- **Rješenje:** 32 bitni registar R[1] ima funkciju odredišta i određen je 5-bitnom kombinacijom u polju *ra.* Registar R[2] je izvor jednog operanda i određen je poljem *rb,* a registar R[28] je izvor drugog operanda i određen je poljem *rc.* Operacijski kod instrukcije add je $(01100)_2$

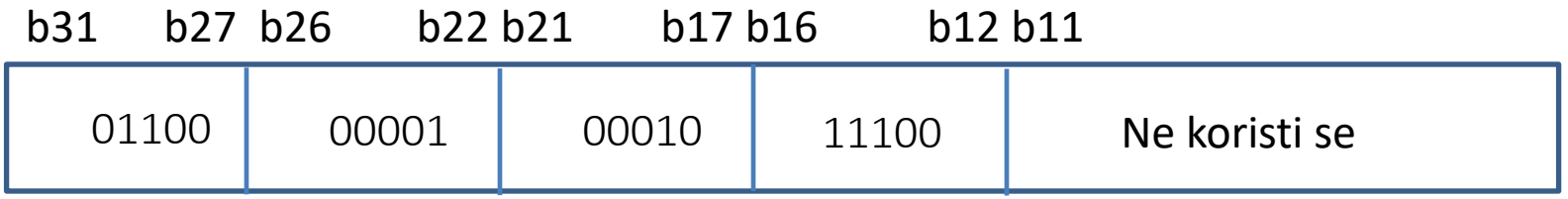


Primjer aritmetičke instrukcije

- **Rješenje:** Instrukcija add r1, r2, r28 određuje operaciju:

  R[1] <- R[2] + R[28]

- Vrijednost pojedinih bitovnih polja za navedenu instrukciju je prikazan na slijedećoj slici:

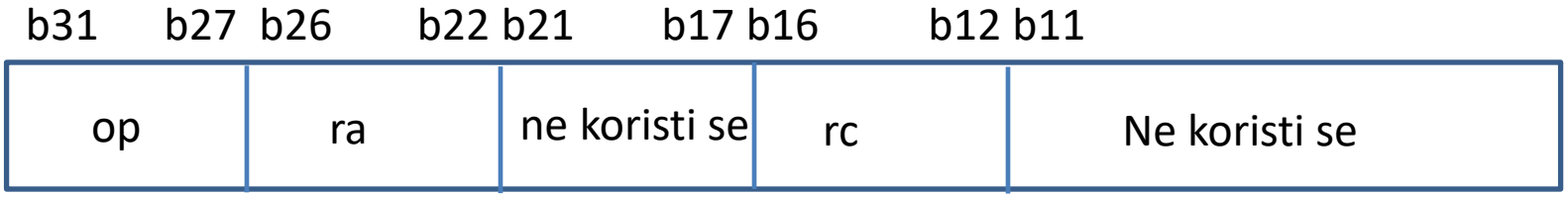


Primjer logičke instrukcije

- **Zadatak:** Za zadani format aritmetičke instrukcije s dvama operandima pojednostavljenog modela 32-bitnog RISC procesora odredite vrijednosti pojedinih bitovnih polja za logičku instrukciju:

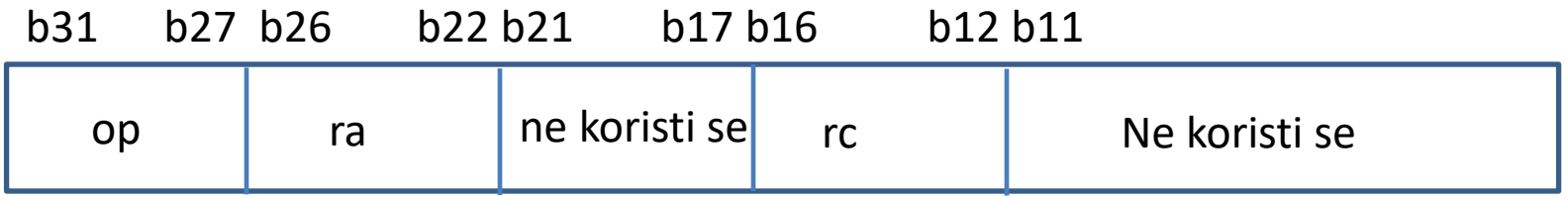
  not r17, r21

- Operacijski kod instrukcije *not* je $(11000)_2$



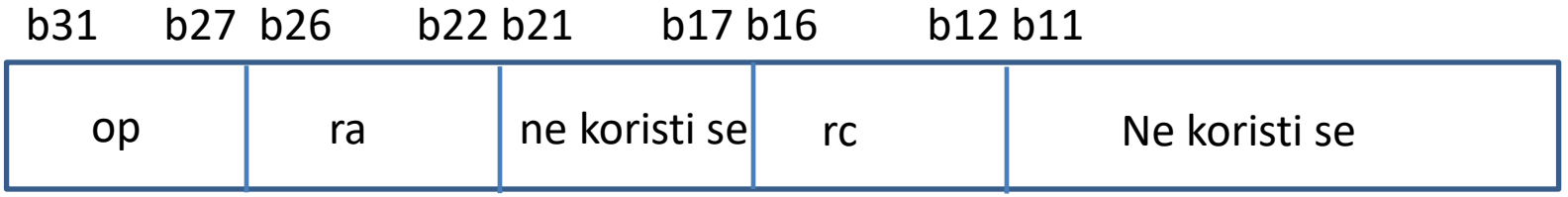
Primjer logičke instrukcije

- **Zadatak:** Uz pretpostavku da su sadržaji registara R[17]= $(00000006)_{16}$ i R[21]= $(FF00FF01)_{16}$ neposredno prije izvođenja instrukcije, odredite sadržaj registara nakon izvođenja instrukcije.



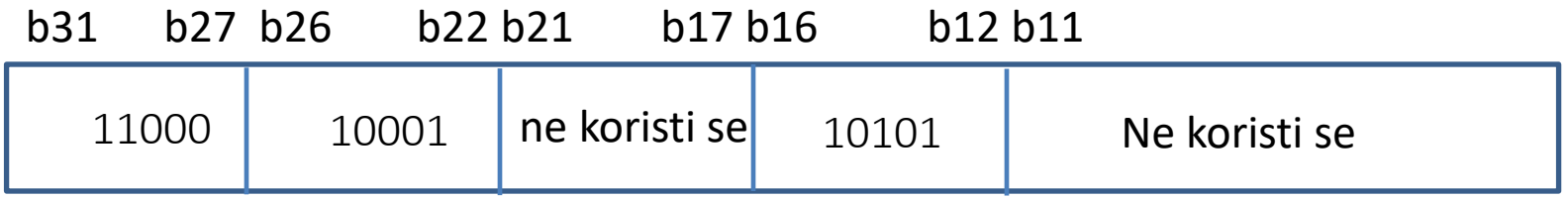
Primjer logičke instrukcije

- **Rješenje:** Logička instrukcija *not ra, rc,* pri čemu je ***rc* izvorišni registar**, a ***ra* odredišni registar** izvodi se tako da se iz izvorišnog registra uzima 32-bitni podatak te se njegova vrijednost komplementira (jedinični komplement) i pohranjuje u odredišni registar.



Primjer logičke instrukcije

- **Rješenje:** Sadržaji registara nakon izvođenja instrukcije *not r17, r21* su:
  - sadržaj izvorišnog registra R[21] se ne mijenja, tj. $R[21]=(FF00FF01)_{16}$
  - sadržaj odredišnog registra $R[17]=(00FF00FE)_{16}$
- Vrijednost pojedinih bitovnih polja za navedenu instrukciju je prikazan na slijedećoj slici:

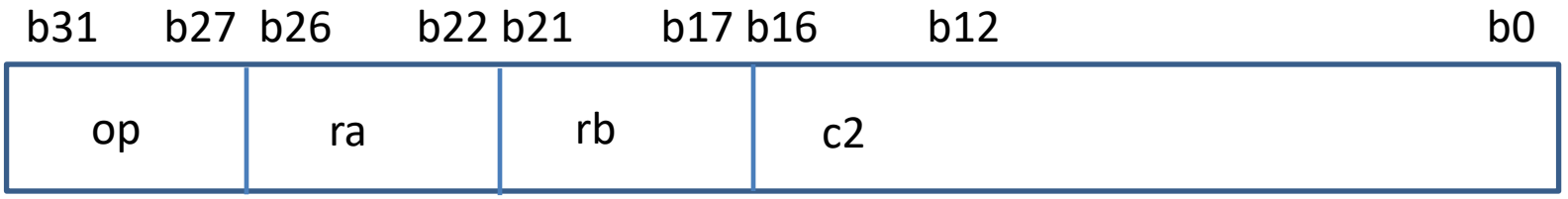


Primjer memorijske instrukcije

- **Zadatak:** Za zadani format memorijske instrukcije *ld (load)* pojednostavljenog modela 32-bitnog RISC procesora, odredite vrijednosti pojedinih bitovnih polja za instrukciju:

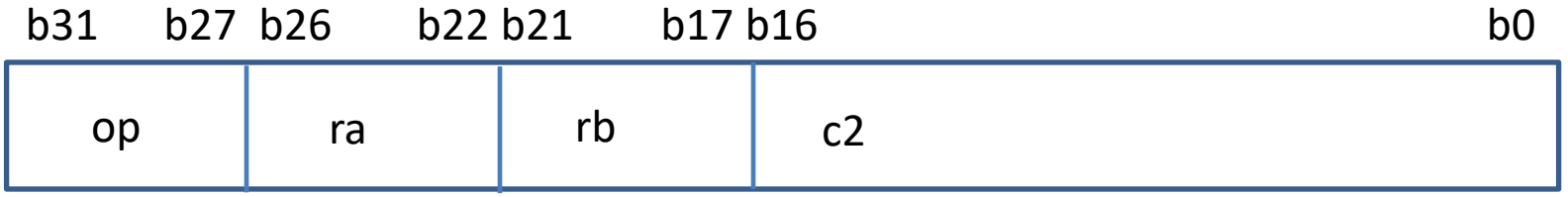
  ld r4, 31

- Operacijski kod instrukcije *ld* je $(00001)_2$



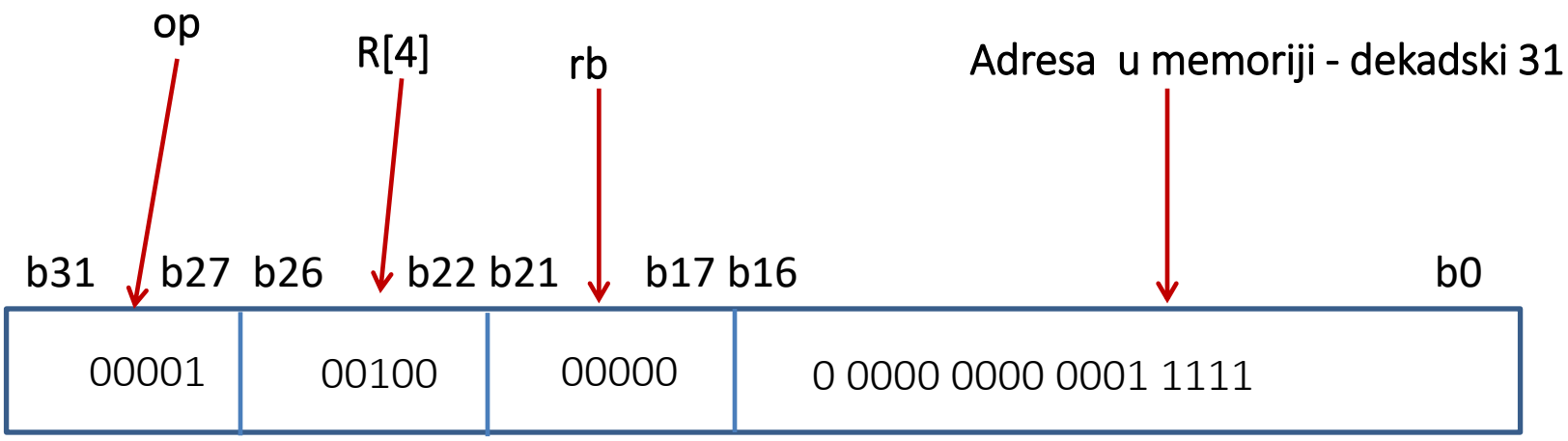
Primjer memorijske instrukcije

- **Rješenje:** Zapis instrukcije *ld r4, 31* podrazumijeva da je polje *rb* jednako $(00000)_2$ (koristi se samo jedan operand *ra* za učitavanje vrijednosti)
- To je signal upravljačkoj jedinici da se adresa memorijske lokacije na kojoj se pohranjuje sadržaj polja *r4* formira u 17-bitnom polju *c2*



Primjer memorijske instrukcije

- **Rješenje:** Vrijednost pojedinih bitovnih polja za navedenu instrukciju je prikazan na slijedećoj slici:

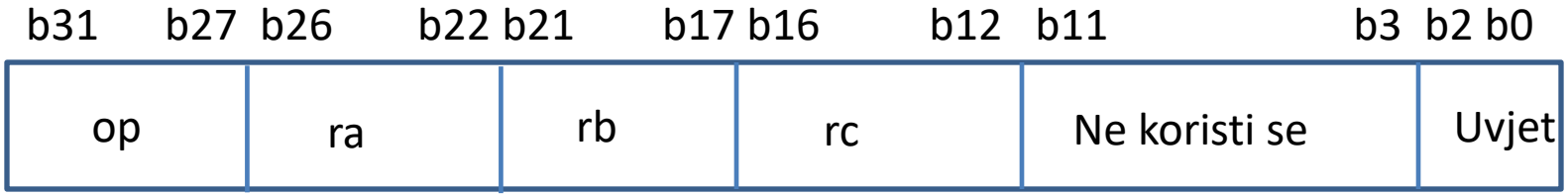


Primjer instrukcije grananja

- **Zadatak:** Odredite promijenjene sadržaje registara nakon izvođenja strojne instrukcije

  brlzr r7,r1,r5

- Navedena instrukcija se nalazi u memoriji računala s početnom adresom $(00\ 00\ 00\ 10)_{16}$
- Uvjet grananja je zadan kao $(010)_2$
- Operacijski kod instrukcije *brlzr* je $(01001)_2$

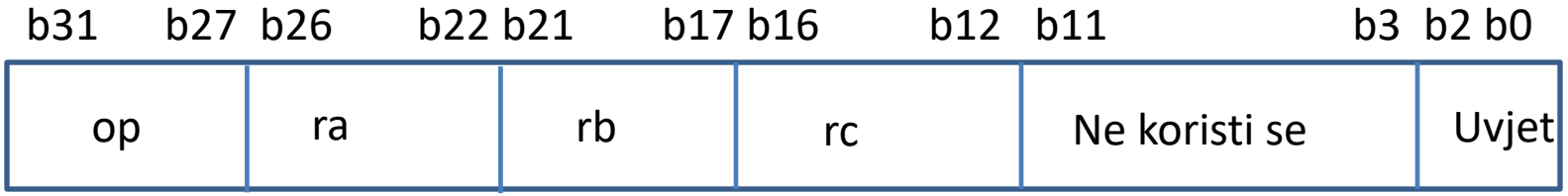


Primjer instrukcije grananja

- **Zadatak:** Sadržaj registra R[1] je $(01\ 00\ 00\ 00)_{16}$
- Odredite sadržaj registra R[7] kada je sadržaj registra R[5]:
    a) $(01\ 00\ 00\ 00)_{16}$
    b) $(01\ 00\ 00\ 00)_{16}$

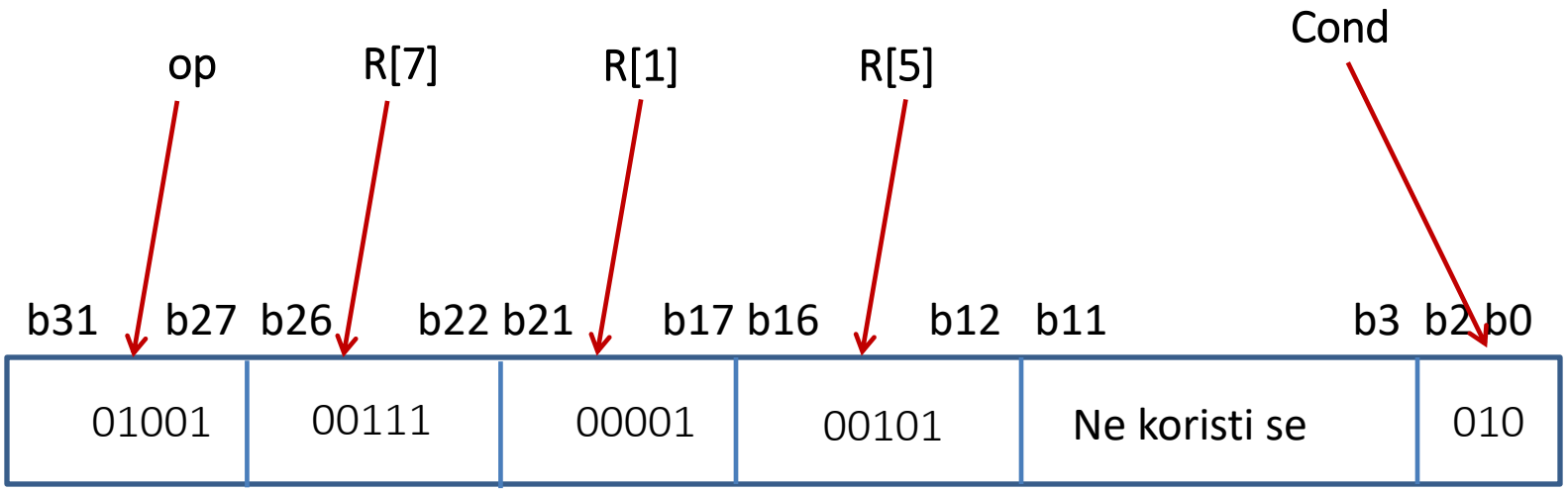
Primjer instrukcije grananja

- **Rješenje :** Zadana instrukcija *brlzr r7,r1,r5* se može čitati na slijedeći način:
  - *Ako je r5 = 0, tada izvršavaj program od instrukcije na adresi R[1]*
  - *Ako je R[5] <> 0, tada izvršavaj program od instrukcije na adresi R[7]*



Primjer instrukcije grananja

- **Rješenje :** Iz zadanih uvjeta zadatka, slijedi format instrukcije kao na niže navedenoj slici.
- Za R[5]=$(01\ 00\ 00\ 00)_{16}$ **-> PC =** R[1]=$(01\ 00\ 00\ 00)_{16}$
- Za R[5]=$(00\ 00\ 00\ 01)_{16}$ **-> PC =** R[7]=$(00\ 00\ 00\ 14)_{16}$

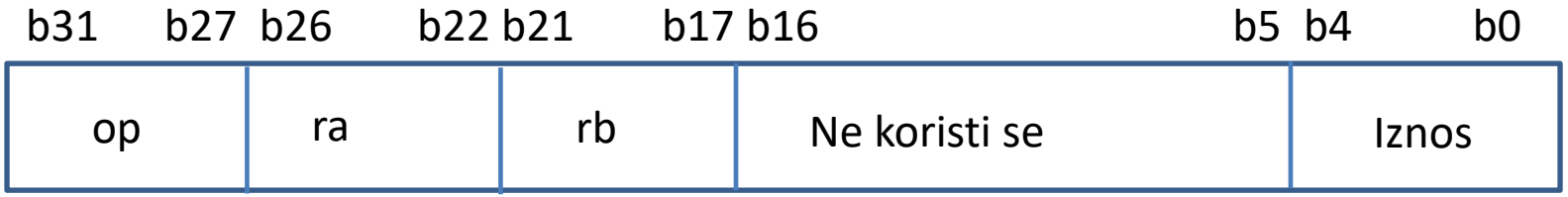


Aritmetičke instrukcije posmaka

- Operatori posmaka (engl. **shift**) služe za pomak svih bitova vrijednosti varijable u lijevo ili u desno
- Posmak bitova u varijabli za jedno mjesto u lijevo odgovara množenju vrijednosti varijable sa 2, dok pomak za jedno mjesto u desno rezultira dijeljenjem vrijednosti varijable sa 2
- Elektronička računala u skupu svojih strojnih naredbi u pravilu imaju naredbe za posmak vrijednosti u registru, i tako izvršeno množenje ili dijeljenje sa višekratnikom broja 2 je bitno brže u odnosu na klasično množenje i dijeljenje
- Broj pomaka je određen parametrom

Aritmetičke instrukcije posmaka

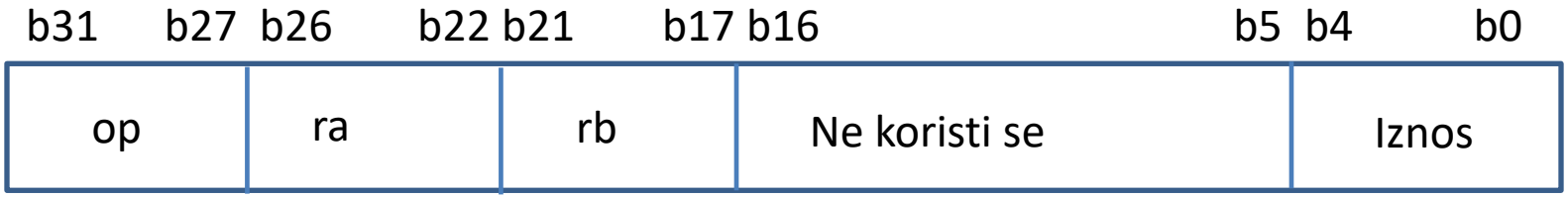
- Format posmačne instrukcije prikazan je na slici:



- 5-bitno polje (pozicije b27 - b31) sadržava operacijski kod kojim se specificira jedna od operacija posmaka (*shr*- posmak udesno, *shra*- aritmetički posmak udesno, *shl*- posmak ulijevo, *shc* - kružni posmak)
- 5-bitno polje ra određuje odredišni registar R[ra] u koji će se smjestiti posmaknuti operand, a 5-bitno polje rb specificira izvor operanda, tj. registar R [rb]
- Pet najmanje značajnih bitova (bO - b4) instrukcije sadržava 5-bitni nepredznačeni cijeli broj kojim se specificira za koliko će se mjesta operand posmaknuti (od 0 - 31 binarnih mjesta)

Primjer aritmetičke instrukcije posmaka

- **Zadatak:** Odredite promijenjene sadržaje registara nakon izvođenja strojne instrukcije *shr r2,r4,4* pojednostavljenog modela 32-bitnog RISC procesora kao na slici:



- ako su početni sadržaji registara R[2] =00 FF FF 00 i R[4] = FF FF FF 80

Primjer aritmetičke instrukcije posmaka

- **Rješenje:** Izvođenjem instrukcije shr r2,r4,4 posmiče se aritmetički operand dohvaćen iz registra R[4] za četiri mjesta udesno **Podsjetimo da posmak za jedno mjesto udesno odgovara dijeljenju s dva !**
- Znači posmak udesno za 4 mjesta odgovara dijeljenju broja sa 16

Primjer aritmetičke instrukcije posmaka

- **Rješenje:** Ako podijelimo vrijednost registra R[4] sa 16 dobiva se:

  R[4] = 0F FF FF F8

- Vrijednost registra R[2] ostaje nepromijenjena
- Format instrukcije je prikazan na slici (za **op** 11010):