# 5. PROJEKTIRANJE PROCESORA SA CJEVOVODOM

## 5.1. Cjevovod

Cjevovod je već spominjano u prethodnim poglavljima kao rješenje kojim se povećavaju performanse procesora. Projektiranje cjevovoda na prvi pogled je zbunjujuće. To je razlog da će se prvo analizirati tehnike projektiranja cjevovoda, zatim će se pristupiti i samom procesu projektiranja. I u ovom slučaju kao primjer poslužiti će jednostavan RISC, SRC.

### 5.1.1. Kratak pregled

Postoji analogija između procesa obrade naredbi u procesoru i izrade nekog proizvoda. U oba slučaja cilj je procesne ili proizvodne potencijale maksimalno iskoristiti. Ovo se postiže podjelom posla na korake, manje cjeline, koje mogu posao raditi paralelno. Što je veći broj koraka na koji je posao podijeljen moguće je postići veću propusnost, odnosno proizvesti više proizvoda u jedinici vremena ili kod računala obraditi veći broj naredbi u jedinici vremena.

**Cjevovod i proizvodna linija**. Principa rada i karakteristike cjevovoda mogu se objasniti usporedbom cjevovoda i proizvodne linije. Kao primjer uzeti će se linija za proizvodnju jednostavnih metalnih dijelova. Proizvodnja se sastoji od bušenja, rezanja, poliranja i pakovanja. Svaki komad ima različito se obrađuje u svakom pojedinom koraku proizvodnog procesa. Ukoliko proces proizvodnje nije uspostavljen, radnik bi uzeo komad materijala, izbušio ga, zatim izrezao, polirao i spakovao. Proces bi radnik ponavljao s novim komadom materijala. Slika 5.1.a prikazuje usporedbu ovog proces i procesa obrade naredbe za zbrajanje. Sa slike je vidljivo da je samo jedna proizvodna (obradne)jedinica zauzeta u jednom vremenskom trenutku, dok ostale proizvodne (obradne) jedinice čekaju. U primjeru samo je obrađuje ALU odnosno samo se buši stražnja ploča. Odmah na prvi pogled uočljiva je slaba efikasnost ovakvog pristupa.

**Zaposliti što više sve funkcionalne jedinice**. Cjevovod kao i organizacija proizvodnog procesa imaju jedinstven cilj, što više zaposliti sve funkcionalne jedinice, odnosno što više skratiti vrijeme koje nisu zaposlene. Na slici 5.1.b. prikazana je izmijenjena situacija kada su sve proizvodne jedinice maksimalno zaposlene. U ovom slučaju pet različitih naredbi se nalazi u različitim stadijima obrade isto kao i u proizvodnom procesu pet različitih dijelova. Radnik na organiziranoj proizvodnoj liniji mora znati što treba napraviti sa svakim različitim komadom kao što procesor mora znati što za svaku naredbom u pojedinom trenutku treba napraviti. Dio programa izvodi se na sljedeći način:

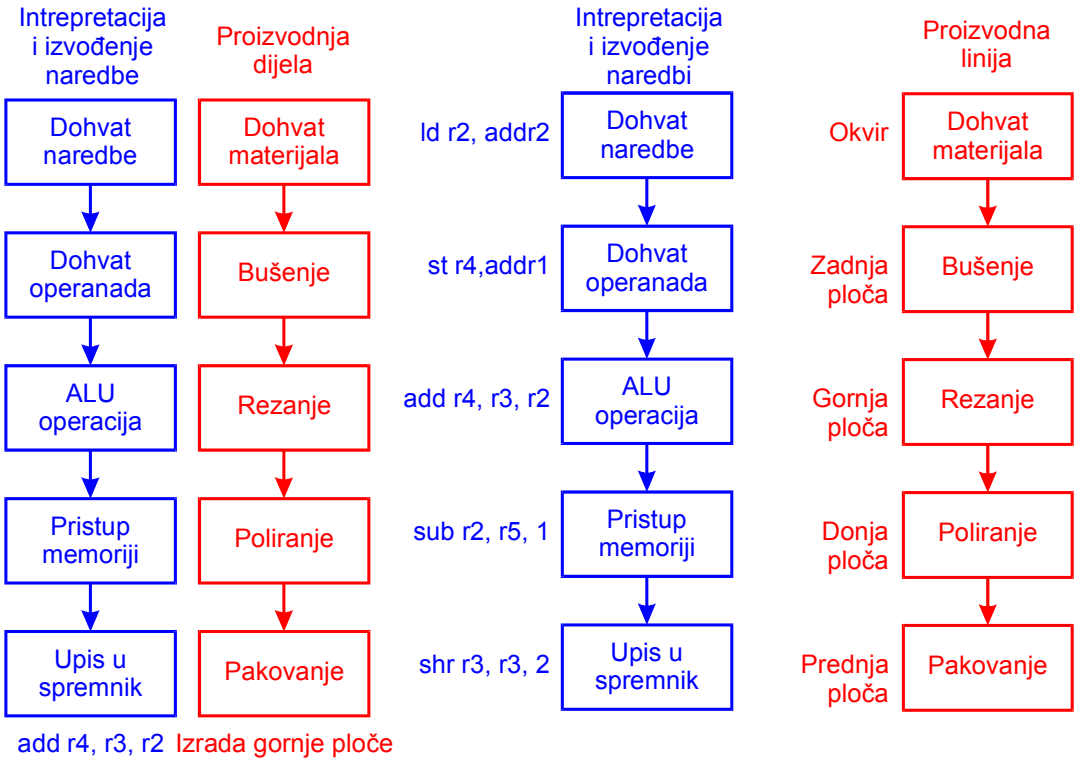
```
shr     r3, r3, 2       ;pohrana rezultata u r3
sub     r2, r5, 1       ;mirovanje – nije potreban pristup memoriji
add     r4, r3, r2      ;izvođenje ALU operacije
st      r4, addr1       ;dohvat r4 i addr1
ld      r2, addr2       ;dohvat naredbe
```

Za primijetiti je da su razine u cjevovodu numerirane odozgo prema dole, a naredbe dolaze obrnuto, odozdo prema gore.

- Naredba shr je prva ušla u cjevovod i ona se trenutno nalazi u fazi upisa rezultata u spremnik r3.
- Naredba sub je u fazi pristupa memoriji, ali kako ova naredba nema ovakvu aktivnost nalazi se u fazi mirovanja. Slično vrijedi u proizvodnom procesu obrade dijelova. Tamo npr. donja ploča ne zahtjeva proces poliranja pa kad taj dio dođe u tu fazu obrade, obrada se jednostavno preskače.

- Naredba add je u trećem koraku obrade, operandi su već prebačeni na ulaze ALU i spremni su za obradu.
- Naredba st je u svom drugom koraku kada se iz dijela naredbe, konstanta c2, izdvaja addr1 te sadržaj r4 priprema se kako bi se mogao u četvrtom koraku prebaciti u memoriju.
- Naredba shr se tek uzima iz memorije.

Sve funkcionalne jedinice su zauzete obradom jedne od naredbi. Ovim pristupom praktički se izvodi jedna naredba u jednom taktu.




a) Obrada bez cjevovoda/proizvodnog procesa b) Obrada sa cjevovodom/proizvodnim procesom

**Propusnost i vrijeme izvođenja naredbe (latentnost)**. Kod tro-sabirničke arhitekture, obrađene u prethodnom poglavlju pokazano je kao za izvođenje naredbe za zbrajanje je potrebno tri takta. Ako se i pretpostavi da se sve naredbe izvode za tri takta slijedi da je propusnost (*bandwidth*) jedna trećina frekvencije takta, tj. ako je takt 100 MHz propusnost je 33 MIPSa. Vrijeme izvođenja naredbe (*latency*) je tri takt ciklusa ili za prethodni primjer 30 ns. Kod procesora sa cjevovodom izvodi se po jedna naredba u svakom taktu, odnosno propusnost je 100 MIPSa, dok se vrijeme izvođenja povećava na pet taktova ili 50 ns.

Može se zaključiti da cjevovod povećava propusnost procesora, dok povećava vrijeme izvođenja naredbe. Povećanje trajanja naredbi utječe samo na programe s izrazito malim brojem naredbi.

**Otežavajući čimbenici**. Kod procesora projektiranog u prethodnom poglavlju tek po završetku izvođenja prethodne naredbe sljedeća se uzima na obradu. Ovim je osigurano da je svaka promjena sadržaja spremnika ili memorijske lokacije obavljena do trenutka kada se započinje s obradom sljedeće naredbe. Ovo nije slučaj kod obrade sa cjevovodom kada se istovremeno obrađuje više naredbi. Nije rijedak slučaj kada

jedna naredba izvodi obradu čiji rezultat je potreban sljedećoj naredbi. Kod cjevovoda postoji mogućnost da prva naredba je u fazi obrade operanada (rezultat još nije pohranjen u odredišni spremnik) a sljedeća naredba je već u fazi dohvata operanada. U ovom slučaju sljedeća naredba ne obrađuje ispravne podatke. U ovakvom slučaju kaže se da su naredbe međuovisne.

Kod programskog odsječka, prikazanog kao primjer obrade pomoću cjevovoda, postoji međuovisnost među naredbama. Naime, naredba shr r3, r3, 2, koja je u fazi upisa rezultata u odredišni spremnik r3 paralelno se izvodi s trećom naredbom add r4, r3, r2 koja je u fazi zbrajanja operanada upisanih u r3 i r2. Prema programskoj logici naredba add bi trebala pribrojiti sadržaj spremnika r3 koji je pomaknut u desno za dva mjesta naredbom shr. Ali kod procesora sa cjevovodom to ne bi bio slučaj. Pribrojila bi se vrijednost koja nije pomaknuta za dva mjesta. Postoje dva rješenja ovakvog problema. Prva mogućnost je detektirati međuovisnost te zabraniti ovisnoj naredbi ulazak u cjevovod dok prva naredba ne završi s obradom. Drugo je rješenje da se rezultat obrade prve ovisne naredbe proslijedi ne čekajući fazu upisa rezultata naredbi koja koristi taj rezultat (*forwarding*).

Sličan, ali nešto složeniji problem postoji prilikom upisa podatka iz memorije u spremnik. U ovom slučaju operand je dostupan tek u četvrtom koraku i ne postoji mogućnost da se on proslijedi prije sljedećim naredbama. Procesor može detektirati ovakvo stanje i zaustaviti obradu sljedećih naredbi dok se podatak ne upiše u spremnik. Postoji mogućnost i programskog rješavanja ovih problema. Program prevodilac mora otkriti međuovisnost naredbi i između njih ubacivati nezavisne naredbe, obično nop, te tako programski odgađati početak izvođenja naredbe.

Uobičajen pristup programskog rješavanja ovog problema kada se programeru izričito naglasi da kod dohvata operanda iz memorije taj je operand dostupan tek nakon n naredbi. Ova restrikcija odnosi se na tehniku pisanja programa ali ne mijenja skup naredbi procesora.

Isti problem susreće se kod naredbi za uvjetno grananje. Naime adresa naredbe koja slijedi iza naredbe za grananje poznata je tek u trenutku kada se izračuna da li je ili nije zadovoljen uvjet grananja. Kod cjevovoda ovo je poznato tek nakon drugog koraka, kada je već sljedeća naredba uzeta u obradu. Procesor bi tada morao ovu naredbu ili odbaciti ili, što je bolje rješenje, program može ovdje umetnuti naredbu koja se mora svakako izvesti bez obzira na naredbu za grananje, odnosno neovisna je o naredbi za grananje. Ova naredba poznata je pod nazivom umetak za kašnjenje grananja (*branch delay slot*).

U oba slučaja postavljaju se dodatni zahtjevi na programera, ili kod programiranja u višim jezicima na programe prevodioce. Ukoliko programer ili program prevodilac ne može pronaći odgovarajuću naredbu koju umeće između ovisnih naredbi koristi se naredba nop.

### 5.1.2. Temeljne pretpostavke kod projektiranja cjevovoda

Ideja cjevovoda pojavila se već 60tih godina sa ciljem povećanja propusnosti računala. Od tada do danas razvijena su brojna sklopovska i programska rješenja cjevovoda kojima se povećava učinkovitost sustava. Nisu sva rješenja jedinstvena, ali u narednim razmatranjima obraditi će se ona najčešće korištena.

**Skup naredbi procesora ostaje nepromijenjen**. Skup naredbi procesora u osnovi ostaje nepromijenjen iako se za njihovu realizaciju koristile različite arhitekture procesora optimizirane sa ciljem postizanja veće brzine obrade. Temeljno pravilo kod

projektiranja cjevovoda je da naredbe moraju izvoditi istu obradu i davati iste rezultate neovisno o arhitekturi na kojoj se izvode. Jedina iznimka može biti uvođenje ovisnosti između naredbi koja rezultira određenim izmjenama u postupku izrade programa.

Formalno, apstraktan opis naredbi, RTN opis, je isti za procesora sa i bez cjevovoda. Ovo pravilo slijedi iz zahtjeva o kompatibilnosti novih procesora sa starim generacijama.

**Izmjene u organizaciji memorije**. Kod realizacije cjevovoda značajno je osigurati da jedna naredba u svakom taktu uđe u cjevovod. Ovaj zahtjev rezultira u izmjenama memorijskog modela prikazanog u prethodnom poglavlju. Naime, nije dozvoljeno da procesor čeka na sporu memoriju kako bi pristupio naredbi ili operandu. Ovo je moguće riješiti samo skupljim brzim memorijama. Drugi problem je što u konceptu cjevovoda svaka naredba u prvom taktu dohvata naredbu a u četvrtom taktu pristupa memoriji. Ovo nije moguće riješiti jednostavnim memorijskim modelom prikazanim u prethodnom poglavlju. Potrebno je postaviti dvije nezavisne memorije, jednu za program, a drugu za podatke te osigurati im istovremeni, paralelni pristup. Ova podjela na zasebnu memoriju za program i za podatke te posebni sabirnički sustav koji osigurava paralelni pristup naziva se Harvardska arhitektura. Ideja je dosta stara i potječe iz 1940 godine, a primijenjena je na računalu Mark III, razvijenom na Harvardskom sveučilištu koji je imao zasebnu memoriju za program i zasebnu za podatke.

**Spremnici opće namjene**. U prethodnom poglavlju pokazalo se učinkovito imati tri sabirnice. Slično vrijedi kod procesora sa cjevovodom. Ovakav pristup zahtijeva da spremnici opće namjene imaju pristup svim sabirnicama, na neke samo pišu a sa drugih čitaju, tzv. spremnici s tri pristupa (*three port register file*). Ovakav pristup omogućava postavljanje istovremeno na dvije sabirnice dva operanda i upis rezultata.

**Izmjena sustava sabirnica i putova podataka**. Cjevovod zahtijeva izmjenu sabirničke strukture. Umjesto povezivanja spremnika preko zajedničke ili zajedničkih sabirnica podaci se direktno prenose iz spremnika u spremnik nezavisnim putovima. Ovakav pristup povezivanja (*point to point*) omogućava istovremeni prijenos većeg broja podataka. Ovo je neophodno kako bi se realiziralo istovremeno više aktivnosti što je i cilj cjevovoda. Ovakav pristup često je jasniji za razumijevanje budući sve veze imaju unaprijed određenu namjenu za razliku od pristupa sa zajedničkim sabirnica kada se struktura mijenja iz takta u takt u ovisnosti o obradi koja se izvodi.

I kod cjevovoda važi pravilo za sabirnice da samo jedan spremnik može određivati u jednom taktu stanje na sabirnici. Ovo rezultira uvođenjem dodatnih spremnika, tzv. spremnika cjevovoda (*pipeline registers*) koji služe prijenosu podataka iz koraka u korak cjevovoda, pa čak i ako se podatak u nekom koraku ne obrađuje. Tako podaci i naredba ili neki njeni dijelovi moraju se propagirati korak po korak kroz cjevovod.

Osim izmjena u broju i funkciji spremnika te načinu njihova povezivanja izmjene se moraju napraviti i kod ALUa. Sve aritmetičke i logičke naredbe moraju se obaviti u jednom taktu. Ovo se posebice odnosi na posmak koji je u prethodnom poglavlju riješen kroz više koraka.

**Dodatno sklopovlje**. Cjevovod obično zahtjeva i dodatno sklopovlje koje ima zadatak dodatno ubrzati rad procesora i pojednostavniti neke operacije. Primjer je nezavisan sklop za inkrementiranje sadržaja programskog brojila. Cilj je rasteretiti ALU ove operacije. Također brojni digitalni multipleksori koriste se za usmjeravanje informacija.

### 5.1.3. Tehnike projektiranja cjevovoda

Zadani i opisani skup naredbi SRCa koristiti će se prilikom projektiranja procesora sa cjevovodom. Radi jednostavnosti izostaviti će se postavljanje u početno stanje, reset, i obrada posebnih stanja i prekida. Naravno, utjecaj posebnih stanja biti će djelomično prodiskutiran.

Kao i kod projektiranja procesora u prethodnom poglavlju isti postupak koristi se kod projektiranja procesora sa cjevovodom. Polazište je skup naredbi procesora te analiza kako pojedine skupine naredbi se preslikavaju na strukturu procesora sa cjevovodom. Ovim se detaljno razrađuje struktura putova podataka i podešava kako bi se kvalitetno i učinkovito izvele naredbe. Nakon što je razrađena struktura putova podataka analiziraju se potrebni upravljački signali kojima se realizira potreban prijenos podataka. Projektiranje okončava razvojem sklopovlja za detekciju međusobne ovisnosti naredbi.

**Podjela naredbi**. Naredbe se mogu, slično kao i u prethodnom poglavlju, podijeliti u ovisnosti kako se podaci prenose preko putova podataka, odnosno kako i gdje se podaci prenose tijekom obrade naredbe. U osnovi postoje tri vrste naredbi, ALU, prijenos podataka (load, store) i naredbe za kontrolu programskog toka ili grananja. Kako bi se bolje shvatilo djelovanje naredbe u njenim pojedinim fazama izvođenja cijeli postupak grafički će se prikazati. Općenito sve naredbe podijeliti će se na pet koraka:

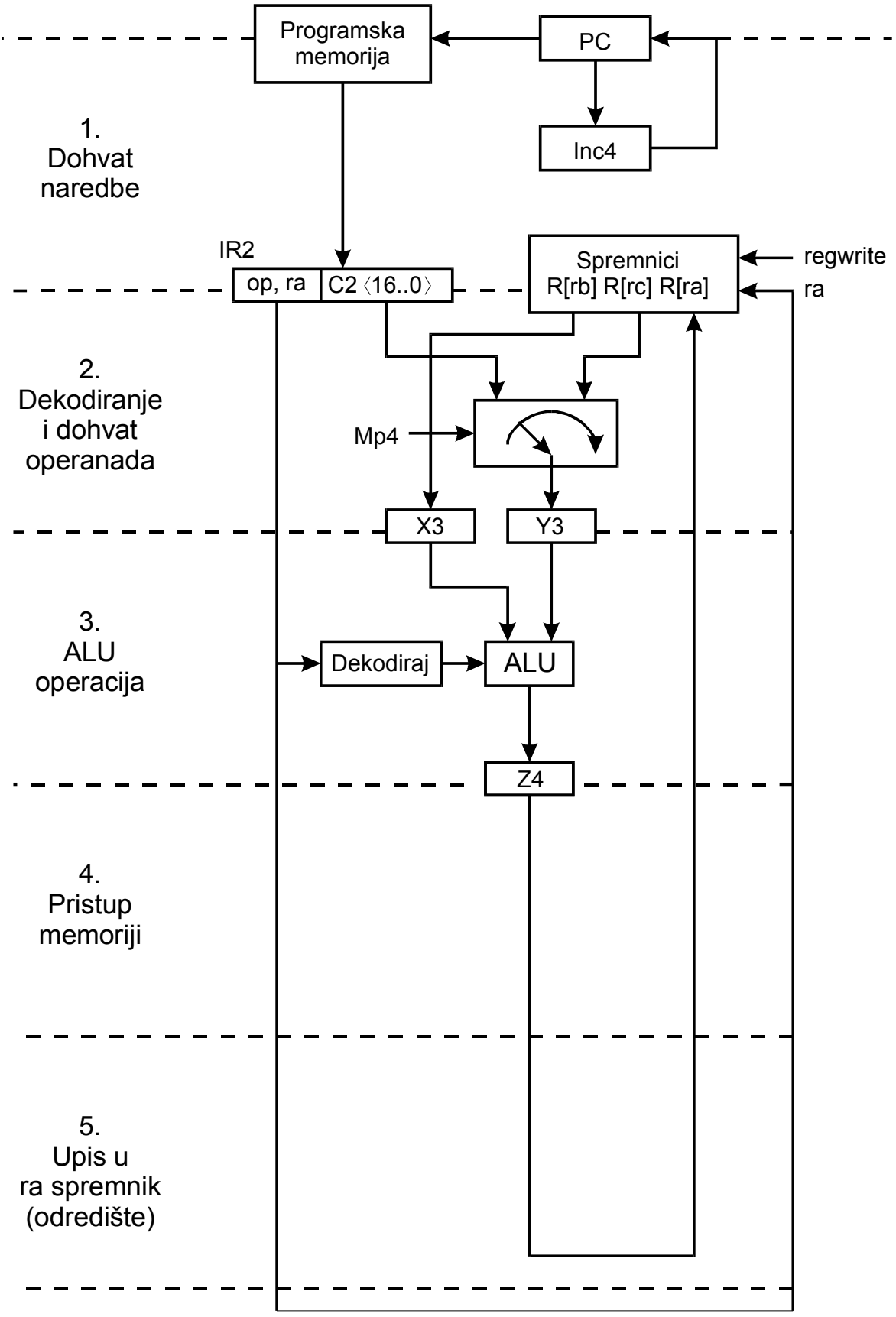
1. dohvat naredbe,
2. dekodiranje i dohvat operanada,
3. ALU operacija,
4. pristup memoriji,
5. upis u spremnik.

Ovih pet koraka koji se susreću tijekom obrade naredbe su pet koraka u cjevovodu kojima su pridruženi odgovarajući spremnici, sabirnice i upravljački signali.

**Propagacija iz koraka u korak**. Slika 5.2. prikazuje pet koraka koji se susreću u izvođenju ALU naredbe. Detalji će biti naknadno objašnjeni. Za primijetiti je da ovo nisu pet dijelova cjevovoda, budući da se analizira i obrađuje samo jedna naredba. Ovdje se radi samo o razbijanju jedne skupine naredbi na pet specificiranih koraka koji će biti jedinstveni cijelom skupu naredbi.

Kod interpretacije slike potrebno je također naglasiti da sinkrono sa sistemskim taktom podaci se prenose iz koraka u korak. Spremnici su numerirani sukladno koraku u kojem se koriste i svi su bridom upravljani (*master-slave flip-flop*). Spremnici prikazani na pri vrhu koraka su ulazni dok oni pri dnu su izlazni spremnici.

**Upravljački signali vezani uz operacijski kod**. Prije prelaska na objašnjenje rada sklopovlja potrebno je specificirati neke upravljačke signale koji upravljaju prijenosom podataka iz koraka u korak cjevovoda ovisno o tipu naredbe koja se izvodi. Neki od upravljačkih signala prikazani su na slici 5.3. Tako npr spremnik naredbe koristi se u drugom koraku pa je označen kao IR2, a temeljen njegovog sadržaja generiraju se signali uvjet i neposredni. Preostali signali koristiti će se u drugim fazama obrade naredbi.




Slika 5.2. Prikaz izvođenja ALU naredbi kod procesora sa cjevovodom.

grananja (*branch*) := br $\vee$ brl:
uvjet (*cond*) := (IR2$\langle$2..0$\rangle$ = 1) $\vee$ ((IR2$\langle$2..1$\rangle$ = 1) $\wedge$ (IR2$\langle$0$\rangle$ $\oplus$ R[rc] = 0)) $\vee$
((IR2$\langle$2..1$\rangle$ = 2) $\wedge$ (IR2$\langle$0$\rangle$ $\oplus$ R[rc]$\langle$31$\rangle$)):

posmak (*sh*) := shr $\vee$ shra $\vee$ shl $\vee$ shla:
alu := add $\vee$ addi $\vee$ sub $\vee$ neg $\vee$ and $\vee$ andi $\vee$ or $\vee$ ori $\vee$ not $\vee$ pomak:
neposredno (*imm*) := addi $\vee$ andi $\vee$ ori $\vee$ (pomak $\wedge$ IR2$\langle$4..0$\rangle$ $\neq$ 0):
čitaj (*load*) := ld $\vee$ ldr:
čitaj_adrese (*ladr*):= la $\vee$ lar:
piši (*store*):= st $\vee$ str:
čitaj_piši (*l-s*) := čitaj $\vee$ $\vee$ piši:
upis_spremnik (*regwrite*) := čitaj $\vee$ čitaj_adrese $\vee$ brl $\vee$ alu:
pomak (*dsp*) := ld $\vee$ st $\vee$ la:
relativno (*rl*) := ldr $\vee$ str $\vee$ lar:

Tablica 5.1. Upravljački signali koji opisuju aktivnosti u cjevovodu.

**ALU naredbe**. ALU naredbe koje uključuju i posmak prikazane se slikom 5.2. Pretpostavka je da se koristi Harvardska arhitektura, odnosno nezavisna memorija za program i nezavisna za podatke. Također uveden je zaseban sklop za inkrementiranje programskog brojila kao nekoliko zasebnih spremnika za privremenu pohranu podataka. Dohvat i izvođenje ALU naredbi, prema slici 5.2, može se opisati na sljedeći način:

**Korak 1.** Naredba na koju pokazuje programsko brojilo dohvata se iz programske memorije, a sadržaj programskog brojila se inkrementira. Na kraju prvog ciklusa Upisuje se naredba u spremnik naredbe IR2 i nova vrijednost u programsko brojilo.

**Korak 2.** Naredba se čita iz spremnika naredbe te se dekodira njen sadržaj. Radi se o ALU naredbi. Ove naredbe imaju sljedeći format:

| | |
|---|---|
| R[ra] $\leftarrow$ R[rb] op R[rc] | obrada sadržaja spremnika |
| R[ra] $\leftarrow$ R[rb] op c2$\langle$16..0$\rangle$ | obrada sadržaja spremnika i konstante |

gdje je iznimka naredba za posmak gdje je broj mjesta za koje se posmak izvodi specificiran ili konstantom c2$\langle$4..0$\rangle$ ili ako je ista jednaka nuli s R[rc]$\langle$4..0$\rangle$. Tako u drugom koraku koji podrazumijeva dohvat operanada dohvata se ili sadržaji R[rb] i R[rc] ili R[rb] i c2. R[rb] se upisuje u privremeni spremnik X3 dok se u privremeni spremnik Y3 upisuje ili R[rc] ili konstanta c2. Pomoću 32 bitovni 2 – 1 multipleksora, Mp4, odabire se koji podatak će se upisati u privremeni spremnik Y3, R[rc] ili c2, što ovisi da li se radi o naredbi koja koristi spremnike ili naredbi koja koristi neposredne operande:

Y3 $\leftarrow$ (neposredno $\rightarrow$ c2: $\neg$neposredno $\rightarrow$ R[rc]): drugi ALU operand

Signal neposredno određuje se iz polja operacijskog koda naredbe prema jednadžbi opisanoj u tablici 5.1 i koristi se za upravljanje multipleksorom Mp4.

**Korak 3.** Naredba se dekodira kako bi se odabrala odgovarajuća ALU operacija. Rezultat se upisuje u privremeni spremnik Z4.

**Korak 4.** ALU naredba ne pristupa memoriji pa nema aktivnosti u ovom koraku.

**Korak 5.** U ovom koraku rezultat iz spremnika Z4 upisuje se u odredišni spremnik R[ra]. U ovom koraku potrebno je imati i vrijednost upisanu u Z4 kao i lokaciju odredišta, odnosno polje ra spremnika naredbe. U ovom koraku potrebno je aktivirati signal upisa u spremnike.

Privremeni spremnici X3, Y3 i Z4 omogućavaju propagaciju informacija iz jednog u drugi korak cjevovoda, a posebno su značajni kada se više različitih naredbi propagira kroz cjevovod.

**Naredbe čitaj (*load*) i piši (*store*)**. Slika 5.3 prikazuje putove podataka kojima se realiziraju naredbe koje prebacuju sadržaj spremnika u memoriju i obratno. To su naredbe ld, ldr, st i str. Naredbe za čitanje i pisanje razlikuju se samo u četvrtom koraku, odnosno fazi pristupa memoriji. Također naredba za upis je neaktivna u petom koraku jer kod ove naredbe ne postoji upis u spremnik. Slijedi ponovni RTN opis ovih naredbi:

| | |
|---|---|
| ld (:= op = 1) $\rightarrow$ R[ra] $\leftarrow$ M[disp]: | upis u spremnik iz memorije |
| ldr (:= op = 2) $\rightarrow$ R[ra] $\leftarrow$ M[rel]: | upis u spremnik iz memorije relativno |
| st (:= op = 3) $\rightarrow$ M[disp] $\leftarrow$ R[ra]: | upis u memoriju iz spremnika |
| str (:= op = 4) $\rightarrow$ M[rel] $\leftarrow$ R[ra]: | upis u memoriju relativno iz spremnika |
| la (:= op = 5) $\rightarrow$ R[ra] $\leftarrow$ disp: | upis u spremnik pomak |
| lar (:= op = 6) $\rightarrow$ R[ra]$\leftarrow$ rel: | upis u spremnik relativni pomak |
| disp$\langle$31..0$\rangle$ := ((rb = 0) $\rightarrow$ c2$\langle$16..0$\rangle$:<br>(rb $\neq$ 0) $\rightarrow$ R[rb] + c2$\langle$16..0$\rangle$: | adresa kod apsolutnog adresnog moda |
| rel$\langle$31..0$\rangle$ := PC$\langle$31..0$\rangle$ + c1$\langle$21..0$\rangle$: | relativna adresa |

**Korak 1.** Dohvat naredbe i inkrementiranje sadržaja programskog brojila koji se upisuje i u spremnike PC i PC2.

**Korak 2.** Dohvat operanda. Ako je relativni adresni mod, tada se sadržaj PC2 i c1 prosljeđuje u spremnike X3 i Y3, a ako je apsolutni adresni mod tada se u iste spremnike prosljeđuje sadržaj spremnika rb i konstanta c2. Upravljanje multipleksorima Mp3 i Mp4 koji ostvaruju funkciju prosljeđivanja odgovarajućeg sadržaja dobiva se temeljen sljedećih jednadžbi:

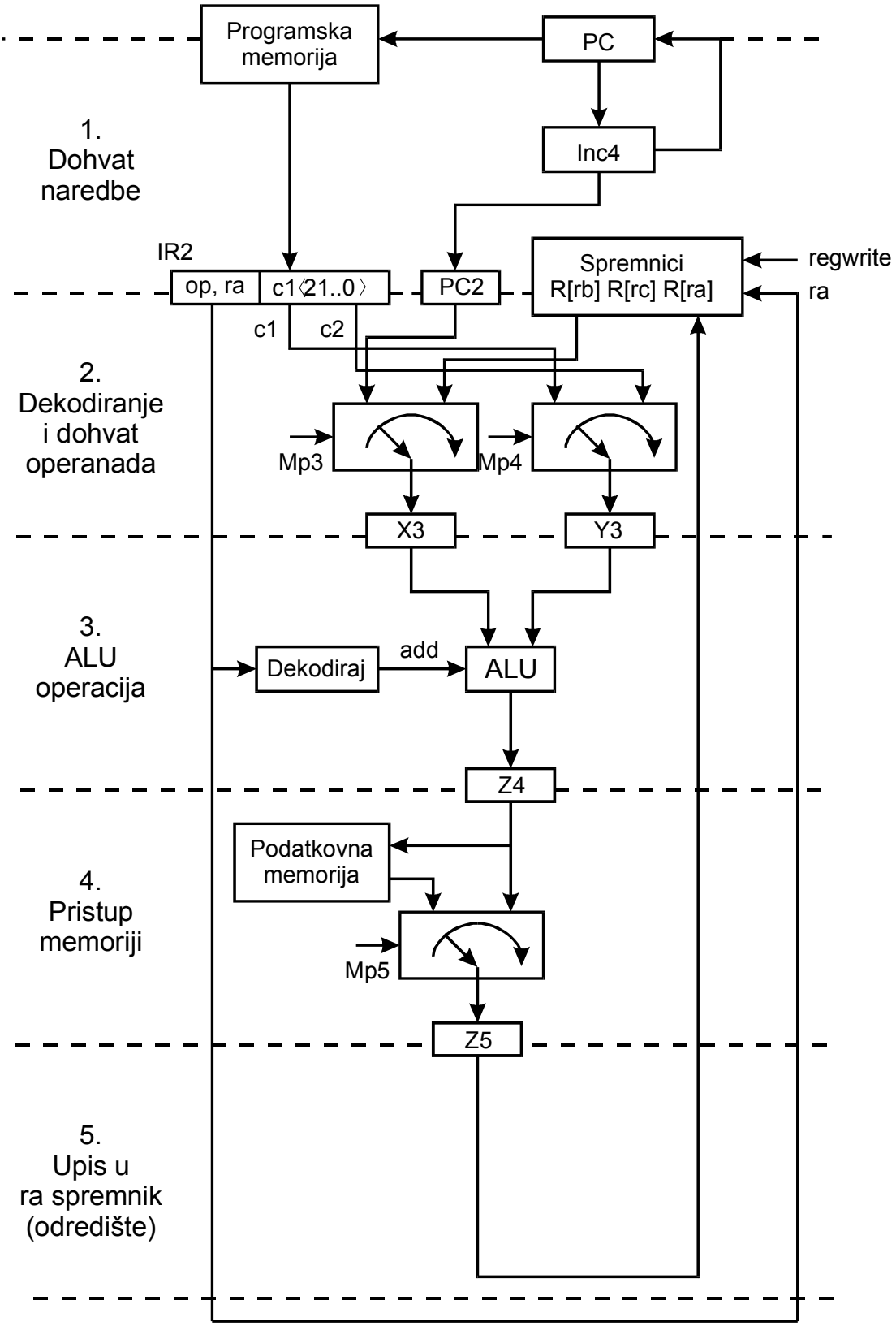
| | |
|---|---|
| X3 $\leftarrow$ (r1 $\rightarrow$ PC: disp $\rightarrow$ R[rb]): | prva komponenta adrese |
| Y3 $\leftarrow$ (r1 $\rightarrow$ c1: disp $\rightarrow$ c2): | druga komponenta adrese |

gdje se ponovo r1 i disp određuju temeljen operacijskog koda naredbe.

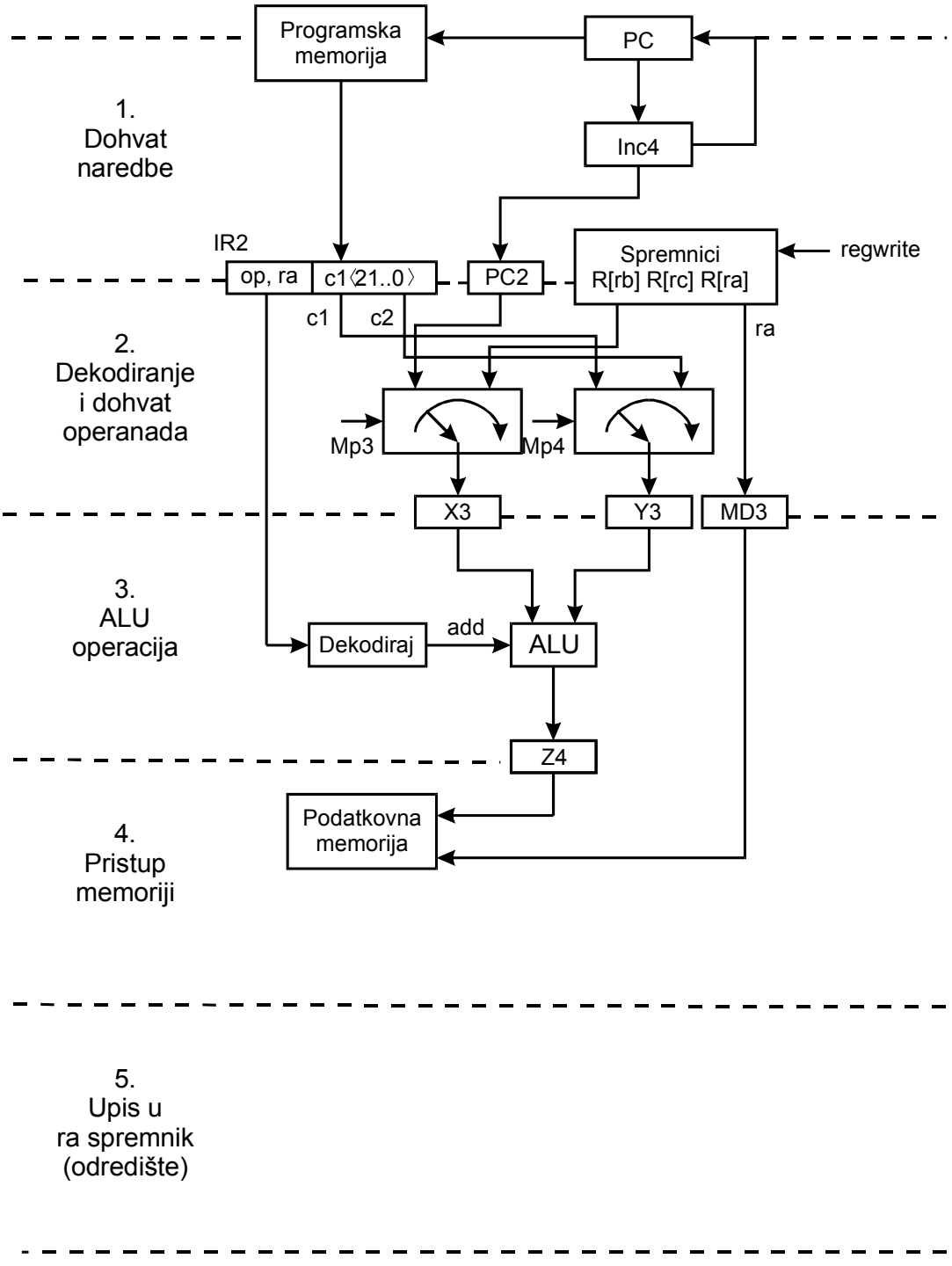
**Korak 3.** Relativna ili apsolutna adresa izračuna se zbrajanjem sadržaja spremnika X3 i Y3. Rezultat se upisuje u spremnik Z4.

**Korak 4.** Ukoliko se radi o čitanju iz memorije, ld ili ldr naredbe, tada se sadržaj iz podatkovne memorije, određen adresom upisanom u Z4 upisuje u spremnik Z5. Ukoliko je upis adrese u spremnik, odnosno naredbe la ili lar, tada se sadržaj spremnika Z4 direktno prebacuje u spremnik Z5. Kod naredbi za upis u memoriju, vrijednost iz spremnika MD3 upisuje se na memorijsku adresu podatkovne memorije određene sadržajem spremnika Z4.

**Korak 5.** Kod svih naredbi za upis u spremnik, sadržaj spremnika Z5 upisuje se u odredišni spremnik ra., određen ra poljem spremnika naredbe IR2. Naredba za upis u memoriju je neaktivna u ovom koraku.




Slika 5.3. Naredba za upis u spremnik adrese ili sadržaja memorijske adrese.




Slika 5.4. Naredba za upis iz spremnika u memoriju.

**Naredbe za grananja**. Naredbe za grananja posebne su utoliko što mijenjaju sadržaj programskog brojila. One su opisane na sljedeći način:

| | | |
|---|---|---|
| uvjet := ( | $c3\langle 2..0\rangle = 0 \rightarrow 0$: | nikada |
| | $c3\langle 2..0\rangle = 1 \rightarrow 1$: | uvijek |
| | $c3\langle 2..0\rangle = 2 \rightarrow R[rc] = 0$: | sadržaj spremnika jednak nuli |
| | $c3\langle 2..0\rangle = 3 \rightarrow R[rc] \neq 0$: | sadržaj spremnika različit od nule |
| | $c3\langle 2..0\rangle = 4 \rightarrow R[rc]\langle 31\rangle = 0$: | sadržaj spremnika pozitivan ili jednak nuli |
| | $c3\langle 2..0\rangle = 5 \rightarrow R[rc]\langle 31\rangle = 1$: | sadržaj spremnika negativan |

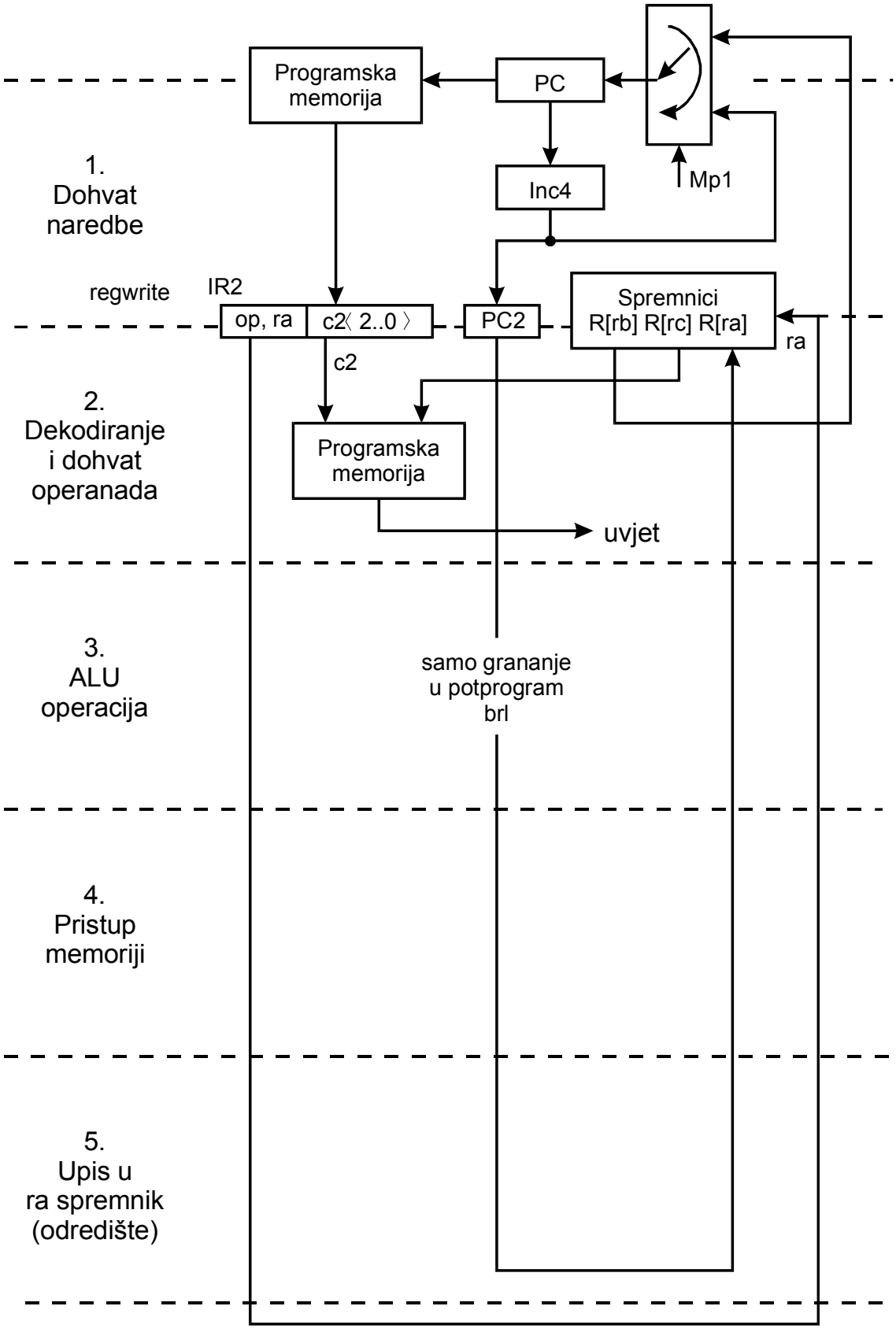
| | |
|---|---|
| br ( := op = 8) $\rightarrow$ (uvjet $\rightarrow$ PC $\leftarrow$ R[rb]): | uvjetno grananje |
| brl ( := op = 9) $\rightarrow$ (R[ra] $\leftarrow$ PC: uvjet $\rightarrow$ PC $\leftarrow$ R[rb]): | uvjetno grananje u potpraogram |

Uvjet je određen temeljen sadržaja prva tri bita spremnika naredbe, $IR\langle 2..0\rangle$, i sadržaja spremnika određenog rc poljem naredbe, R[rc]. On određuje da li će se zamijeniti sadržaj programskog brojila sadržajem spremnika određenog rc poljem naredbe, R[rc]. Slika 5.5 prikazuje ponašanje ove klase naredbi. Kod prethodnih naredbi sadržaj PCa i PC2 je jednak. Kod naredbi za grananje može biti različit što ovisi da li uvjet grananja zadovoljen ili ne. Kod naredbi koje se koriste za grananje u potprogram, brl, potrebno je sačuvati adresu povratka upisom u spremnik ra. Kod ovih naredbi upis sadržaja programskog brojila u spremnik ra izvodi se bez obzira da li je zadovoljen uvjet grananja. Multipleksor Mp1 koji određuje da li se u programsko brojilo upisuje prethodni sadržaj uvećan za četiri ili sadržaj spremnika R[rb] upravljan je prema sljedećem izrazu:

uvjet(IR2, R[rc]) $\rightarrow$ PC $\leftarrow$ R[rb]: X3 $\leftarrow$ PC2.

**Ukupno stanje procesora**. Sadržaji spremnika opće namjene, programskog brojila te svih privremenih spremnika u određenom trenutku određuju sveukupno stanje procesora, odnosno informacije o stanju svake pojedine naredbe u cjevovodu. Sveukupno stanje procesora ne može se povezati s jednom naredbom nego s nizon naredbi koje trenutno procesor obrađuje.

**Skup naredbi**. Provedenim razmatranjima praktički se analizirane sve naredbe, osim naredbe stop. Provedene analize pokazale su da se sve naredbe mogu podijeliti na pet jedinstvenih faza uz određene izmjene u sklopovlju i upravljačkim signalima. Prikazani modeli svake pojedine grupe naredbi vrijede za njihovo izvođenje ne uzimajući u obzir da se može u obradu uzeti druga naredba prije njenog završetka. Slijedi nadogradnja sklopovlja i upravljačkih signala kako bi se moglo izvoditi istovremeno, ali u različitim fazama više naredbi.




Slika 5.5. Naredbe za grananja.

### 5.1.4. Projektiranje putova podataka kod cjevovoda

Informacije vezane uz naredbu i njeno trenutno stanje izvođenja moraju se paralelno propagirati s naredbom kroz cjevovod, kroz tzv. spremnike cjevovoda.

**Dodavanje sklopovlja i upravljačkih signala za podršku realizacije cjevovoda**. Proces projektiranja nakon što su razrađene pojedine skupine naredbi nastavlja se dodavanjem i nadogradnjom potrebnog sklopovlje, spremnika cjevovoda i multipleksora. Ovaj proces znatno je pojednostavljen grupiranjem naredbi. Bez ovog međukoraka bilo bi potrebno zasebno analizirati svaku pojedinu naredbu. Ovaj proces nastavlja se pažljivom analizom slika 5.2 do 5.5. te određivanjem koje se informacije moraju iz jedne faze prenositi u sljedeću.

**RTN opis i projektiranje cjevovoda**. Slijedi opis spremnika cjevovoda kao i RTN opis upravljačkih signala i toka podataka kroz cjevovoda. Slika 5.6. prikazuje spremnike cjevovoda kao i RTN opis toka naredbi kroz cjevovod. Ova slika predstavlja temelj projektiranja SRCa sa cjevovodom. Ona je kombinacija specifikacije putova podataka (spremnika i njihovih veza), te specifikacije aktivnosti određenih pojedinim naredbama prikazanim na slikama 5.2 do 5.5. Broj spremnika cjevovoda moguće je minimizirati korištenjem istih spremnika u određenom koraku za više tipova naredbi obično uz trošak uvođenja multipleksora.

Može se primijetiti kako su upravljački signali označeni prema koraku u kojem se određuju. Jednadžba:

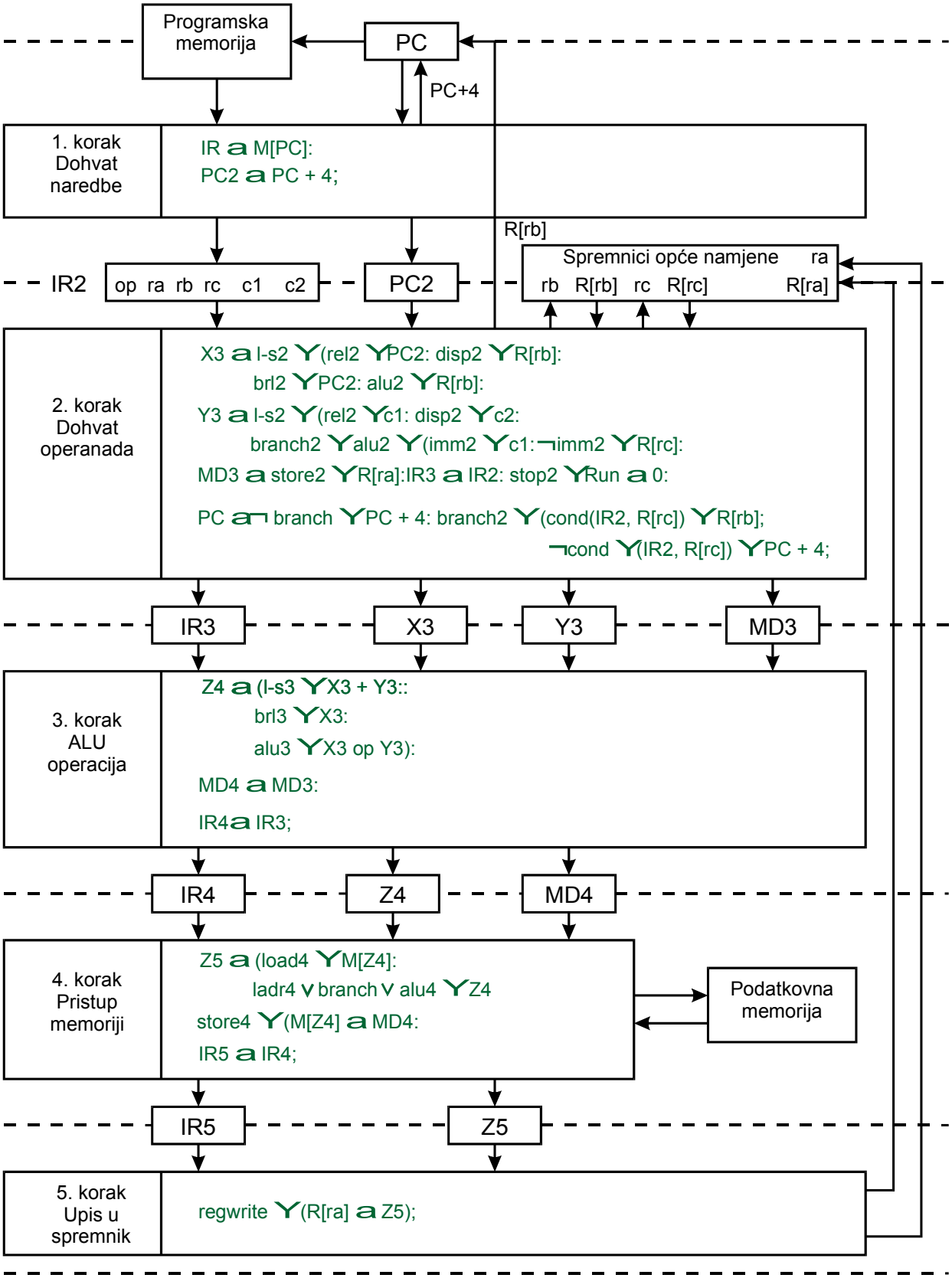
$$PC \leftarrow (\neg branch2 \rightarrow PC + 4 : branch2 \rightarrow (cond(IR2, R[rc]) \rightarrow R[rb] : \neg cond(IR2, R[rc]) \rightarrow PC + 4)) :$$

prikazana u drugom koraku opisuje promjenu sadržaja programskog brojila. Ako se ne radi o naredbi za grananje u drugom koraku, tada se programsko brojilo inkrementira sklopovski u prvom koraku. Ako je u drugom koraku dekodirana naredba za grananje, preko sadržaja spremnika naredbe IR2, i ako je uvjet grananja određen s prva tri bita spremnika naredbe, $IR2\langle 2..0\rangle$, zadovoljen, tada u programsko brojilo upisuje sadržaj spremnika rb, R[rb]. Pažljivom analizom slike 5.6. i njenom usporedbom sa slikama 5.2 do 5.5. može se zaključiti da se svaka naredba izvodi u cjevovodi na isti način kao i da cjevovod nije prisutan.

Sadržaj spremnika naredbe, IR2, mora se prenositi iz koraka u korak kako bi sve informacije o naredbi bile dostupne u svakom koraku. Broj uz IR označava u kojem je koraku taj spremnik implementiran. Također za primijetiti je da se ne koriste u svakom koraku sva polja spremnika naredbe. Tako npr. u drugom koraku, prilikom dekodiranja naredbe, potrebne su skoro sve informacije, dok već u sljedećim koracima samo operacijski kod i polje ra. U trećem koraku operacijski kod potreban je da bi se odredilo o kojoj se aritmetičkoj operaciji radi. U četvrtom koraku operacijski kot neophodan je naredbama za pristup memoriji kako bi se odredilo da li se radi u upisu u memoriju ili čitanju iz memorije. U posljednjem koraku potrebno je ra polje kako bi se odredio spremnik u koji je potrebno upisati rezultat operacije, ukoliko je upis uopće potreban što specificira operacijski kod.

Z4, izlazni spremnik ALUa, može sadržati osim rezultata aritmetičke ili logičke operacije i memorijsku adresu ako se radi o naredbama za pristup memoriji, proslijeđeni sadržaj programskog brojila kod naredbe za grananje u potprogram (programsko brojilo se inkrementiralo u prvom koraku i potrebno ga je upisati spremnik ra).

Spremnik Z5 sadrži:



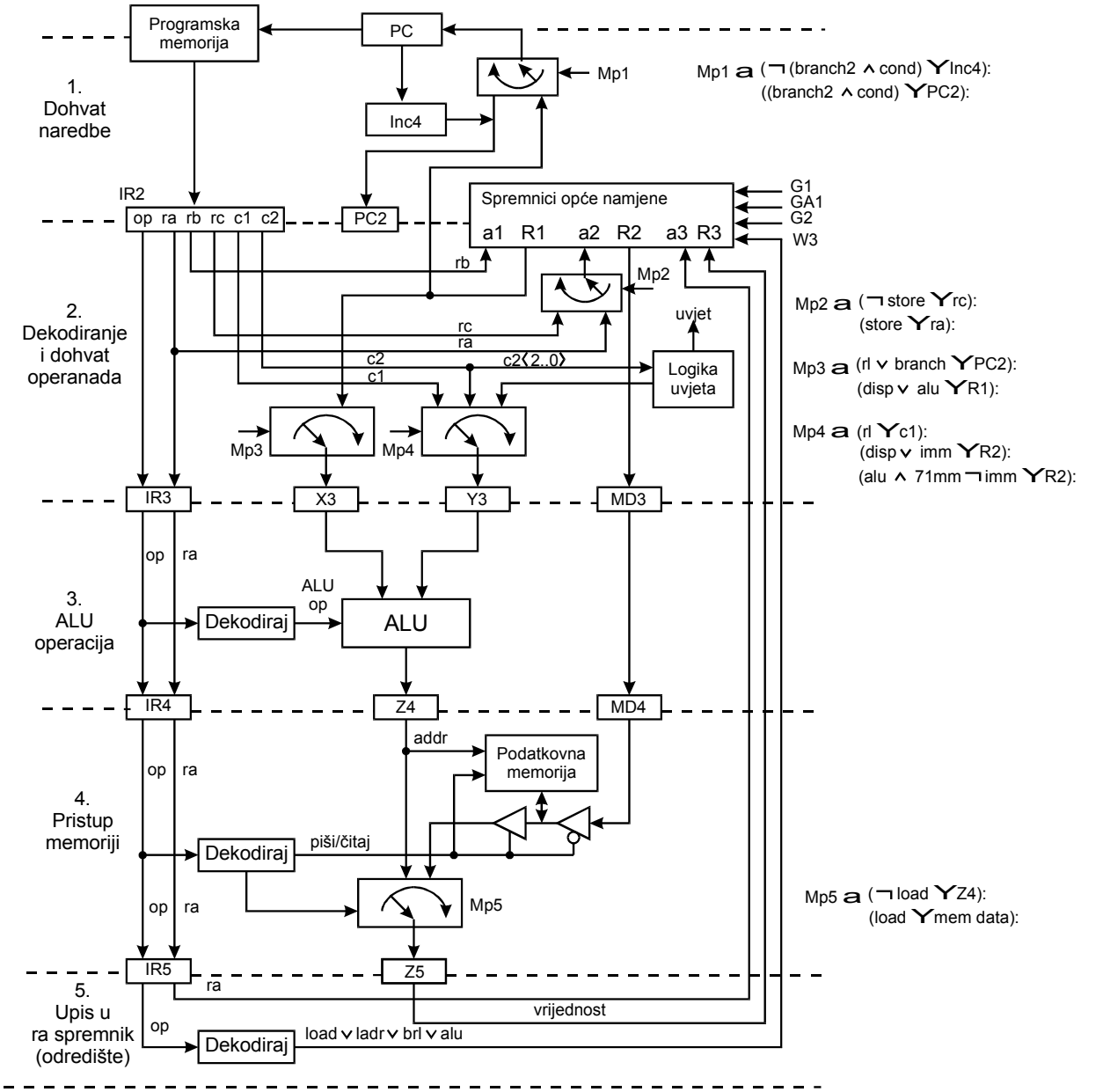

Slika 5.6. Spremnici cjevovoda i RTN opis aktivnosti.

Spremnik Z5 sadrži:

- adresu ukoliko se radi o naredbama za upis adrese u spremnik, la ili lar,
- sadržaj memorijske lokacije kod naredbi za upis sadržaja memorije u spremnik, ld i ldr,
- povratnu adresu kod naredbe za grananje u potprogram,
- rezultat ALU operacije kod ALU naredbi.

Konačno, sadržaj MD3 potrebno je proslijediti u sljedeći korak, u MD4, kako bi podatak bio pripravan za upis u memoriju.

Slika 5.7. prikazuje sklopovlje potrebno za implementaciju funkcija opisanih slikom 5.6.




Slika 5.7. Procesor sa cjevovodom, putovi podataka i upravljački signali.

Na slici 5.7. napravljena je izmjena u oznakama vezanim uz skup spremnika opće namjene. Oznaka a, a1, a2 i a3 predstavlja adresu spremnika, adresne ulaze, dok oznaka R je vezana uz sadržaj spremnika i označava podatak koji se uzima iz ili upisuje u odabrani spremnik. Ovo je bilo potrebno iz razloga što naredba za upis u memoriju koristi ra polje za selektiranje spremnika čiji sadržaj se upisuje u memoriju, pa je u drugom koraku potrebno sadržaj spremnika ra, R[ra] upisati u MD3. Ostale naredbe u drugom koraku koriste samo polja naredbe rb i rc. Slika 5.7 prikazuje koja polja naredbe se prosljeđuju na adresne ulaze skupine spremnika opće namjene. Sadržaj spremnika R1 i R2 postavlja se na sabirnicu signalima G1, GA1 i G2. Signal GA1 ima istu funkciju kao i signal $BA_{out}$ kod procesora bez cjevovoda, odnosno postavlja na izlaz R1 sve nule. Upis u spremnike upravlja se signalom W3, upis preko ulaza R3 u spremnik selektiran adresom na a3.

Kako je već naznačeno multipleksori su označeni s Mp1 –Mp5 i koriste se za prespajanje više ulaza na jedan izlaz. Izrazi kojima se oni upravljaju napisani su na slici 5.7. Npr. multipleksor Mp4 ima tri ulaza, c1, c2 i R2. Na prvi ulaz dovedeno je c1 polje naredbe, IR2, i ono se prespaja se na ulaz spremnika cjevovoda Y3 ukoliko je u drugom koraku jedna od naredbi koja koristi relativno adresiranje, ldr, str i lar. Na drugi ulaz dovedeno je c2 polje naredbe, IR2, i ono se prespaja na ulaz spremnika cjevovoda Y3 ukoliko se radi o naredbama koje koriste neposredno adresiranje, addi, andi i ori, ili ukoliko se radi o naredbama koje koriste adresiranje s pomakom (disp), ld, st i la, ili se radi o naredbi za posmak kod koje je posljednjih pet bita naredbe, c2⟨4..0⟩, različito od nule. Na treći ulaz multipleksora dovodi se sadržaj spremnika sa izlaza R2 i on se prespaja na ulaz spremnika cjevovoda Y3 ukoliko se radi o alu operacijama koje ne koriste konstantu unutar naredbe ili kod naredbe za posmak kada je posljednjih pet bita naredbe, c2⟨4..0⟩, jednako nuli.

Kod svih naredbi polja naredbe rb i rc određuju operande koje se očitava u drugom koraku, a rezultat se u petom koraku upisuje u odredišni spremnik određen poljem naredbe ra, osim kod naredbe za upis sadržaja spremnik u memorijsku lokaciju. Kod ove naredbe, polje ra određuje spremnik čiji sadržaj je potrebno upisati u memoriju. Sadržaj spremnika se u na početku trećeg koraka prebacuje u spremnik cjevovoda MD3. Funkciju odabira spremnika bilo poljem naredbe rc ili ra rješava se pomoću multipleksora Mp2, koji prespaja jedno ili drugo polje naredbe na adresni ulaz spremnika opće namjene a2 u ovisnosti o kojoj se naredbi radi.

**Generiranje upravljačkih signala**. Za razliku od monolitne upravljačke jedinice opisane u prethodnom poglavlju, generiranje upravljačkih signala mora se propagirati paralelno s naredbom kroz cjevovod. Ovo ne bi smjelo predstavljati problem s obzirom da naredba nosi sa sobom sve potrebne informacije iz koraka u korak. Nadalje, većina upravljačkih signala skoro je jedinstvena uz neznatne razlike svim naredbama u pojedinim koracima, kao što je npr. upis u programsko brojilo u prvom koraku. Na slici 5.7. prikazani su upravljački signali multipleksora te signali kojima se čita iz i piše u spremnike opće namjene. Naravni i svi ostali spremnici cjevovoda moraju imati upravljački signal kojim se upisuje u njih sadržaj s ulaza. Za primijetiti je da su spremnici direktno međusobno vezani ili preko multipleksora. Iz tog razloga nije potrebno generirati signale postavljanja sadržaja spremnika na sabirnicu, odnosno spremnici ne moraju biti realizirani s izlazima s tri stanja.

RTN opis na slici 5.7. dostatan je za opis većine upravljačkih signala kojima se upravlja prijenosom podataka. Naravno postoje i neki posebni slučajevi koji će se naknadno obraditi.

### 5.1.5. Tok niza naredbi kroz cjevovod

Kroz sljedeća razmatranja opisati će se na primjeru način tok naredbi kroz cjevovod. Primjer je sljedeći programski odsječak:

```
100:    add   r4, r6, r8          ;R[4] ← R[6] + R[8]
104:    ld    r7, 128(r5)         ;R[7] ← M[R[5] + 128]
108:    brl   r9, r11, 001        ;PC ← R[11](=512):R[9] ← PC
112:    str   r12, 32             ;M[PC+32] ← R[12]

. . . . .

512:    sub   . . . . .           ;sljedeća naredba
```

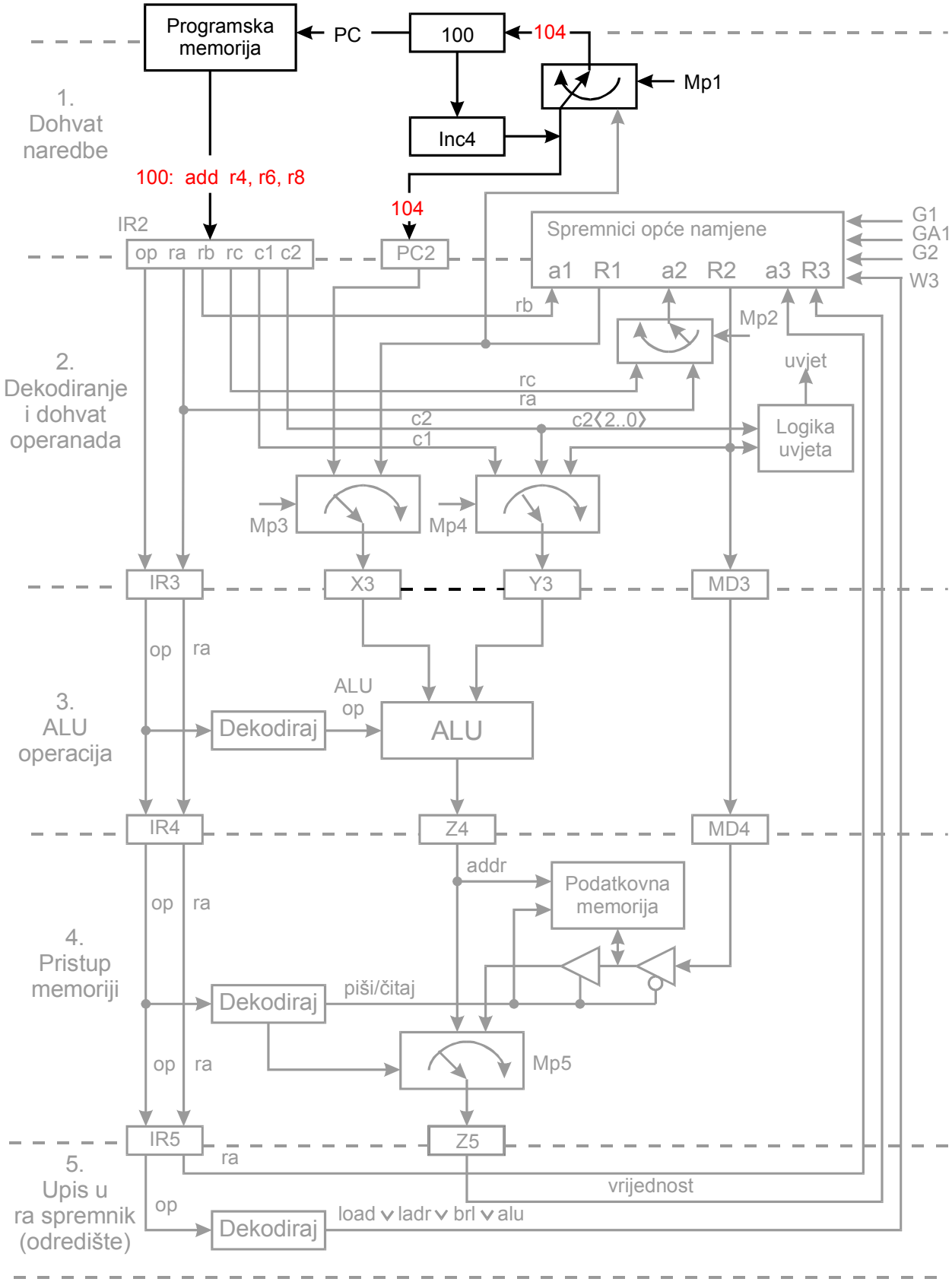
Slike 5.8 do 5.12 opisuju tok naredbi kroz cjevovod, počevši dohvatom naredbe za zbrajanje i završavajući naredbom za oduzimanje. Pretpostavka je da je programsko brojilo inicjalizirano na vrijednost 100, te da spremnici cjevovoda imaju na slici upisane trenutne vrijednosti.

Aktivna stanja u pojedinim trenutcima prikazana su tamnijom bojom od ostalih dijelova putova podataka koji se u pojedinom taktu ne koriste. Na slici su prikazane i vrijednosti upisane u spremnike cjevovoda kao i vrijednosti postavljene na pojedine sabirnice. Pažljivo analizirajući ove slike važno je uočiti da nezavisnost između koraka, odnosno da dolazak nove naredbe ne ometa ispravno izvođenje naredbi uzetih u obradu.

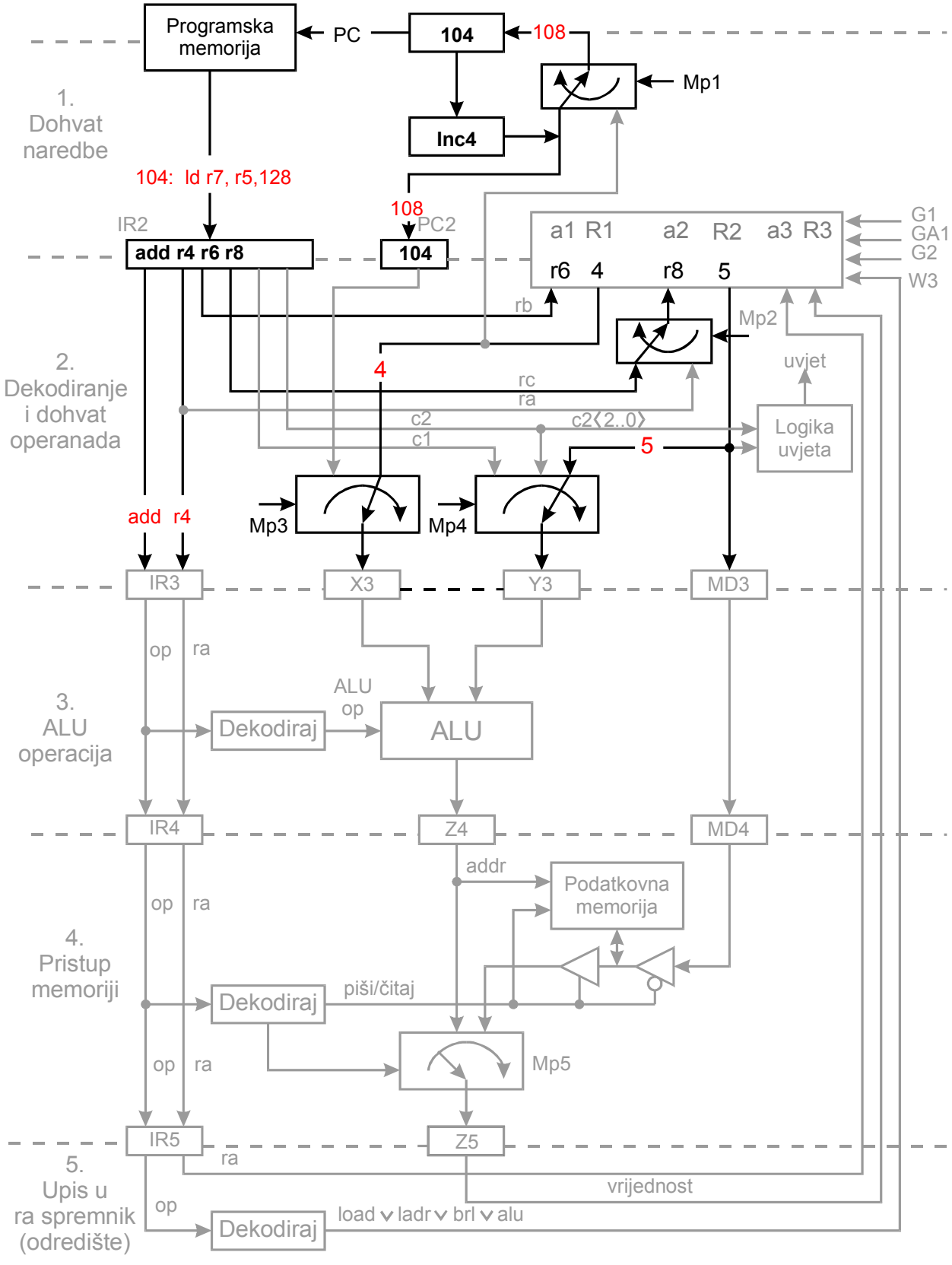
**Prvi takt ciklus: naredba add ulazi u cjevovod**. Na slici 5.8 programsko brojilo pokazuje na adresu 100, adresu programske memorije na kojoj je upisana naredba za zbrajanje. Programska memorija postavlja svoj sadržaj na sabirnicu, a istovremeno procesor inkrementira sadržaj programskog brojila i dovodi novu vrijednost, 104, na njegov ulaz. Slika prikazuje da naredba i nova vrijednost programskog brojila čekaju sljedeći takt kako bi bili upisani u IR2, PC i PC2.

**Drugi takt ciklus: add prelazi u drugi korak; ld ulazi u cjevovod**. Slika 5.9 prikazuje stanje nakon pojave drugog takt impulsa. Naredba za zbrajanje prelazi u drugi korak, dekodiranje sadržaja naredbe. Naredba se upisuje u spremnik naredbe, IR2, kao i sadržaj programskog brojila, 104, u PC2. Dekodiranjem naredbe određeni su operandi. To su sadržaji spremnika r6 i r8, odnosno vrijednosti 4 i 5 koji se iz spremnika preko mulitleksora Mp3 i Mp4 postavljaju na ulaze spremnika cjevovoda X3 i Y3. Operacijski kod, add, i spremnik u koji je potrebno upisati rezultat, r4, postavljeni su na ulaz spremnika cjevovoda IR3. Ovim je prva naredba pripravna za treći korak njenog izvođenja

Druga naredba koju je potrebno izvesti, ld, upisana je na adresi 104 programske memorije. Postavljanjem sadržaja programskog brojila na adresnu sabirnicu programske memorije nareda se dohvata i pripravna je za upis u spremnik naredbe IR2. Ujedno je sadržaj programskog brojila inkrementiran na vrijednost 108 i pripravan je za upis u spremnike IR i IR2.




Slika 5.8. Prvi takt izvođenja programa: naredba add ulazi u cjevovod.




Slika 5.9. Drugi takt izvođenja programa: naredba add ulazi u drugi korak; naredba ld ulazi u cjevovod.

**Treći takt: add prelazi u treći korak; ld u drugi korak; brl ulazi u cjevovod**. Na slici 5.10. prikazano je stanje kada naredba add prelazi u treći korak. Njeni operandi pohranjeni su u spremnike cjevovoda X3 i Y3, a operacijski kod, add, i odredišni spremnik, r4, u IR3. Logika za dekodiranje signalizira ALU da se radi o naredbi za zbrajanje. Kombinacijska logika zbraja sadržaje s ulaza i rezultat operacije se dovodi na ulazu spremnika cjevovoda Z4. Operacijski kod, add, kao i odredišni spremnik r4, spremni su za upis u spremnik cjevovoda IR4.

Naredba ld je ušla u drugi korak u kojem se ona dekodira. Adresa operanda kod ove naredbe određena je sadržajem spremnika rb i konstante c2. U konkretnom slučaju odabire se spremnik r5 čiji sadržaj 16 dovodi se preko multipleksora Mp3 na ulaz spremnika cjevovoda X3. Istovremeno se iz naredbe odabire konstanta c1, 128, te se ista preko multipleksora Mp4 dovodi na ulaz spremnika cjevovoda Y3. Operacijski kod, ld, kao i odredišni spremnik r11 spremni su za upis u spremnik cjevovoda IR3.

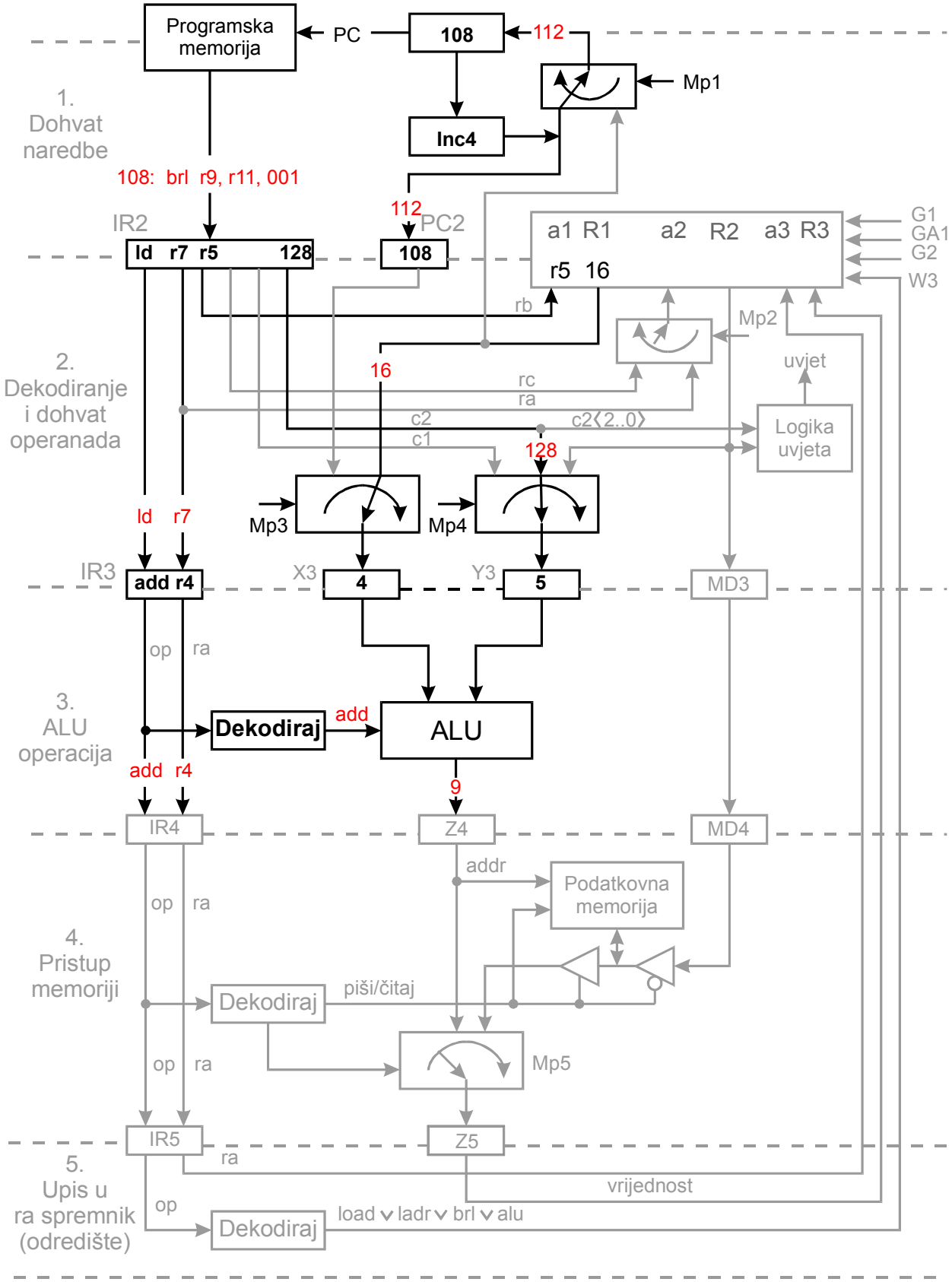
Programsko brojilo pokazuje na adresu 108 programske memorije na kojoj se nalazi naredba brl. Programska memorija naredbu postavlja na izlaz i ona je pripravna za upis u spremnik naredbe IR2. Istovremeno se programsko brojilo inkrementira i pokazuje na novu naredbu upisanu na adresi 112. Ova vrijednost će se u sljedećem taktu upisati u spremnike PC i PC2.

**Četvrti takt: add prelazi u četvrti korak: ld u treći: brl u drugi: str ulazi u cjevovod**. Slika 5.11. prikazuje četvrti takt izvođenja programa. Zbrajanje je izvedeno i rezultat je upisan u privremeni spremnik cjevovoda Z4. Kako se radi o ALU operaciji, koja ne pristupa podatkovnoj memoriji, dekoder samo prespaja izlaz spremnika Z4 na ulaz spremnika Z5.

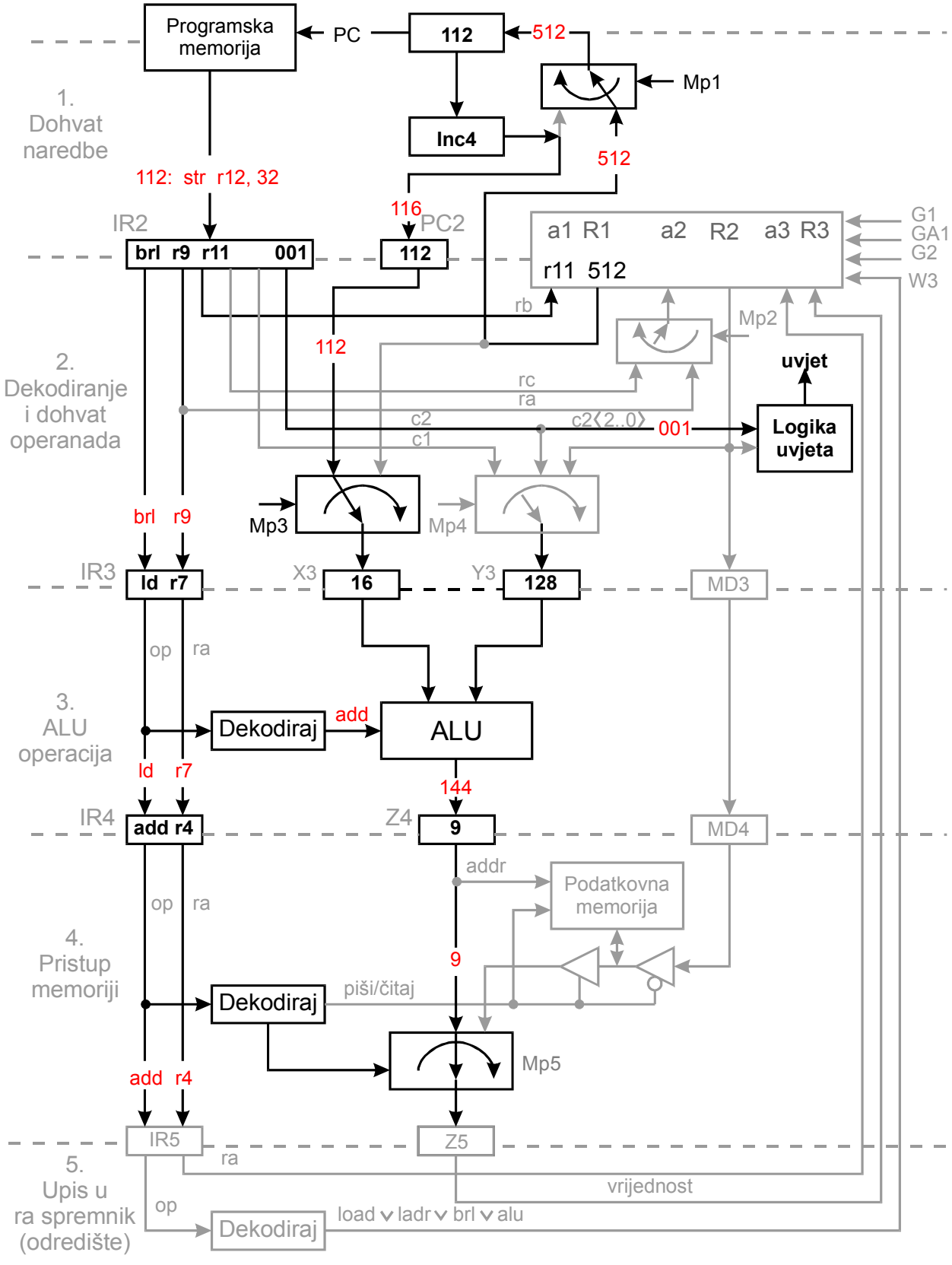
Naredba ld je u trećem koraku u kojem se izračunava adresa podatkovne memorije iz koje je sadržaj potrebno prebaciti u odredišni spremnik, r7. Adresa podatka dobiva se zbrajanjem sadržaja spremnika X3, sadržaj spremnika r5 koji je 16 i konstante c1 koja je 128. Rezultat, 144, čeka sljedeći takt impuls kako bi se upisao u spremnik cjevovoda Z4.

U drugom koraku je naredba za bezuvjetno grananje u potprogram, brl. Adresa potprograma upisana je u spremnik r11, a povratna adresa će se upisati u spremnik r9. Uvjetni kod 001 specificira da se radi o bezuvjetnom grananju. Ovaj kod se preko c2 sabirnice dovodi na logiku uvjeta koja svoj izlaz, uvjet, postavlja u jedinicu. Kako je uvjet jednak jedinici i radi se o naredbi za grananje, multipleksor Mp1 preusmjerava sadržaj spremnika r11, odnosno adresu prve naredbe potprograma, na ulaz programskog brojila. Sada se na ulazu programskog brojila nalazi vrijednost 512, a ne prethodna vrijednost povećana za četiri, koja će se upisati u programsko brojilo sljedećim takt impulsom.

Dok procesor određuje adresu grananja, procesor je već u procesu dohvata sljedeće naredbe. Programsko brojilo pokazuje na adresu 112 programske memorije, na kojoj je upisana naredba str. Naredba se dovodi na ulaz spremnika naredbe, IR2. Ujedno sadržaj programskog brojila se inkrementira te se vodi na ulaz spremnika cjevovoda PC2, ali ne i na ulaz programskog brojila na koji je već multipleksor Mp1 prespojio izlaz spremnika r11. Ova naredba je već ušla u cjevovod i izvesti će se bez obzira na prethodne naredbe. Ovim primjerom objašnjena je potreba odgađanja dohvata naredbe koja se nalazi iza naredbe za grananje ukoliko je ta naredba ovisna o rezultatu grananja. U tom slučaju ubacuje se fiktivna naredba koja ne izvodi nikakvu obradu, nop.




Slika 5.10. Treći takt izvođenja programa: naredba add ulazi u fazu izvođenja ALU; naredba ld ulazi u korak dekodiranja, a naredba brl je u fazi dohvata naredbe.




Slika 5.11. Četvrti takt izvođenja programa: naredba add je izvedena i čeka upis u odredišni spremnik; naredba ld ulazi u korak proračuna memorijske adrese, naredba brl je u fazi dekodiranja i određivanja adrese grananja, a naredba str je u fazi dohvata.

**Peti korak: naredba za zbrajanje završava; naredba ld dohvata podatak iz podatkovne memorije; naredba brl premješta se u treći korak; naredba str dohvata operande; dohvata se prva naredba potprograma sub**. U petom taktu naredba za zbrajanje, add, završava izvođenje upisom rezultata u odredišni spremnik. Rezultat operacije, 9, i adresa spremnika, r4, prosljeđuje se na ulaze spremnika opće namjene. Za primijetiti je da se rezultat u odredišni spremnik upisuje tek na kraju petog koraka. Na početku koraka rezultat operacije se prebacuje u spremnik cjevovoda Z5.

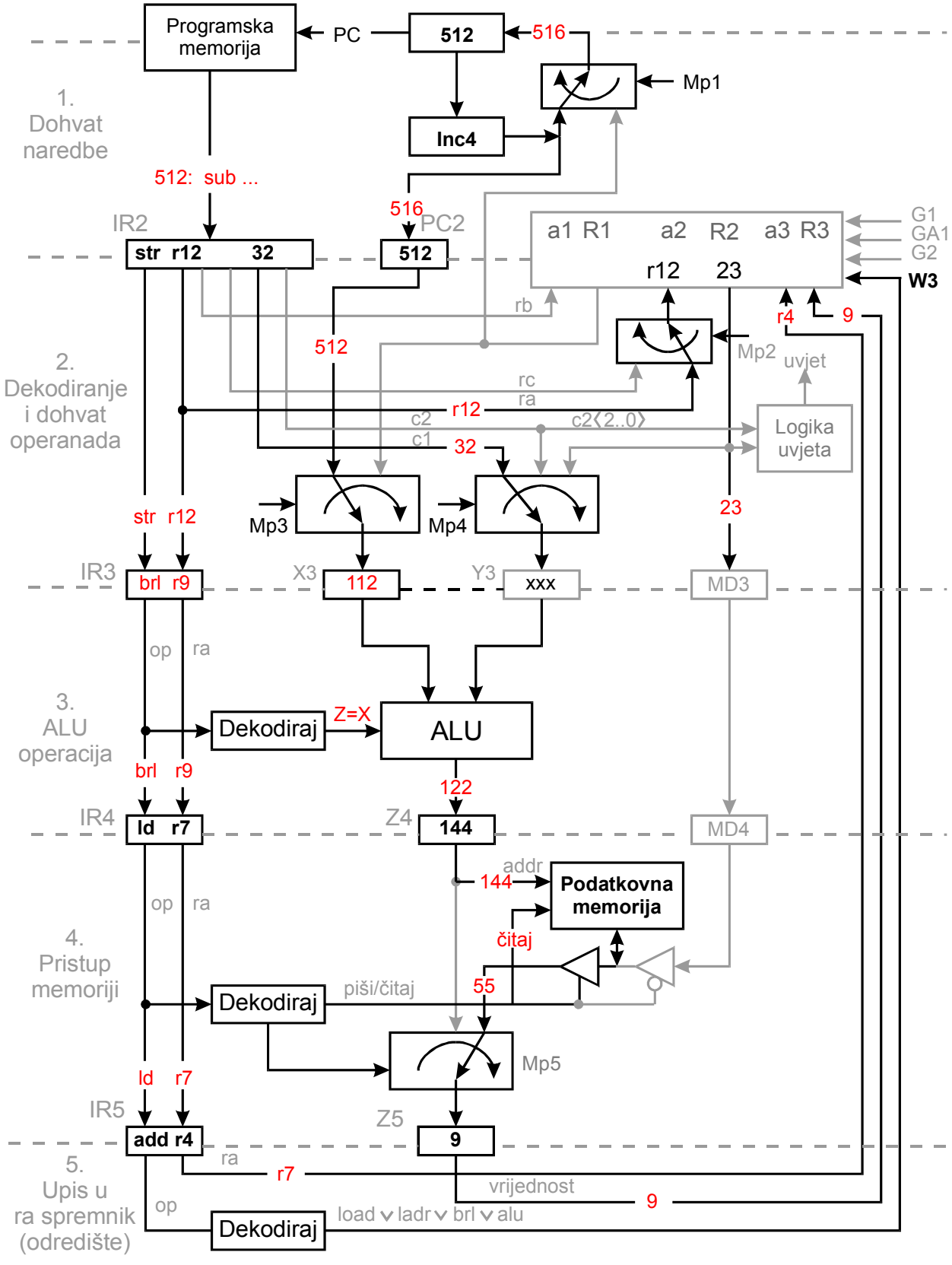
Naredba ld je u četvrtom koraku pristupa podatkovnoj memoriji. Proračunata adresa operanda, 144, postavlja se na adresnu sabirnicu. Logika dekodira da se radi o naredbi za čitanje podatka iz memorije te generira podatkovnoj memoriji signal čitaj. Ujedno multipleksor Mp5 prespaja izlaz memorije na ulaz spremnika cjevovoda Z5. Na njegovom ulazu je sada vrijednost 55. Ujedno, informacija o operacijskom kodu ove naredbe i odredišnom spremniku prosljeđuje se na ulaz spremnika IR5.

Naredba za grananje prelazi u treći korak. Kako ova naredba ne zahtjeva nikakav proračun, kombinacijska logika samo prosljeđuje adresu povratka iz potprograma, 112, prema sljedećem koraku, odnosno spremniku Z4. Ovo se realizira dekodiranjem operacijskog koda naredbe i izdavanjem naloga ALU da izvede naredbe Z = X.

U fazi dekodiranja naredbe i dohvata operanada nalazi se naredba str. Ova naredba koristi relativno adresiranje, M[PC + 32] $\leftarrow$ R[r12]. Zato je bilo potrebno sačuvati inkrementiranu vrijednost programskog brojila iz prethodnog takta. Multipleksor Mp3 prosljeđuje sadržaj PC2 na ulaz spremnika cjevovoda X3 dok multipleksor Mp4 prosljeđuje sadržaj konstante c1 iz PC2 na ulaz spremnika cjevovoda Y3. Ovim je napravljena priprema za proračun memorijske adrese. Kako se radi o naredbi koja upisuje sadržaj spremnika u podatkovnu memoriju, multipleksor Mp2 prespaja polje naredbe ra na adresni ulaz spremnika opće namjene. Sadržaj odabranog spremnika, r12, koji iznosi 23 postavlja se na ulaz spremnika MD3, koji će u sljedećem koraku za vrijeme proračuna adrese samo biti proslijeđen prema spremniku MD4 iz kojega će se prebaciti na izračunatu memorijsku adresu.

Prva naredba potprograma, sub, nalazi se u fazi dohvata naredbe koji je već više puta opisivan.

Peti takt izvođenja programa prikazan je slikom 5.12.




Slika 5.12. Peti takt izvođenja programa: naredba add se završava upisom u odredišni spremnik; naredba ld ulazi u korak čitanja podatka iz podtkovne memorije, naredba brl je proslijeđena kroz ALU fazu, naredba str je u fazi dekodiranja i određivanja adrese grananja, a naredba sub je u fazi dohvata.

### 5.1.6. Opasnosti vezani uz primjenu cjevovoda

Opasnosti vezane uz primjenu sasvim su predvidljivi događaji. Posljedica su izvođenja naredbe ovisne o rezultatu jedne od prethodnih naredbi čija obrada još nije završila. Sukladno navedenom, pogreške koje nastaju zbog primjene cjevovoda rezultat su ili loše napisanog programa prevodioca ili programske pogreške u simboličkom programu, a posljedica su ili nepoznavanja principa rada cjevovoda ili nemarnosti programera. Program prevodilac mora unaprijed predvidjeti, prije izvođenja programa, moguće izvore pogrešaka te ih otkloniti. Pri tome se rukovodi strategijom najgoreg slučaja (*worst case*). Rješenja koja se tada primjenjuju sigurno nisu optimalna u smislu brzine izvođenja programa. Još veći problemi su ukoliko programer sam mora rješavati posebnosti cjevovoda, što je posljedica ljudskog pristupa problemu pri čemu se neke činjenice lako previde.

Opasnosti vezane uz primjenu cjevovoda mogu se podijeliti prema tipu naredbe uz koji su vezani. Iako postoji više mogućih podjela, osnovna je na opasnosti vezane uz podatke i one vezane uz grananja.

**Opasnosti u primjene cjevovoda vezane uz podatke**. Opasnost primjene cjevovoda vezana uz podatke odnosi se na slučaj kad naredba pristupa spremniku ili memorijskoj lokaciji prije nego je u nju neka od prethodnih naredbi upisala rezultat. Kod SRCa koji se analizira podatke mijenjaju alu naredbe, čitaj/piši naredbe i naredba za grananje u potprogram.

Može se analizirati sljedeći programski odsječak:

```
100:    add   r0, r2, r4
104:    sub   r3, r0, r1
```

Naredba za zbrajanje add zbraja sadržaje spremnika r2 i r4 i rezultat upisuje u spremnik r0. U sljedećoj naredbi rezultatu prethodnog zbrajanja oduzima se sadržaj spremnika r1. Opis obrade naredbi sa cjevovodom, slike 5.8. do 5.12, pokazao je da se rezultat operacije upisuje u odredišni spremnik tek u petom koraku, dok se operandi dohvaćaju veću drugom koraku. Tako naredba za oduzimanje pristupa spremniku r0 prije nego je prethodna naredba za zbrajanje u njega upisala rezultat. Ovo se naziva opasnost primjene cjevovoda vezan uz pristup podacima.

Ukoliko se ne primjeni odgovarajuće sklopovsko rješenje ovog problema postavlja se pitanje što je potrebno učiniti kako bi se riješi navedeni problem. Za prikazani programski odsječak naredba za zbrajanje, add, upisuje rezultat u odredišni spremnik r0 tek u petom koraku. Naredba koja koristi rezultat zbrajanja smije najdalje doći do prvog koraka cjevovoda.

Pažljiva analiza slika kojima je opisan prolaz naredbi kroz cjevovod pokazuje da je rezultat operacije zbrajanja dostupan već u trećem koraku, odnosno dostupan je u trenutku kada ga trena sljedeća naredba, naredba za oduzimanje sub. Ovo razmatranje ukazuje na mogućnost konstrukcije sklopovlja koje može pravovremeno proslijediti potreban podatak sljedećoj naredbi, tzv. sklopovlje za prosljeđivanje (*forwarding hardware*).

Prilikom određivanja svih mogućih opasnosti vezanih uz pristup podacima kod cjevovoda potrebno je analizirati sve naredbe i njihove međuovisnosti. Na prvi pogled ovo izgleda složen zadatak, ali moguće ga je znatno pojednostavniti grupiranjem naredbi prema načinu kako pristupaju operandima i kako rezultat upisuju u odredišni

spremnik. Prvo će se analizirati međuovisnost dviju ALU naredbi, zatim preostalih naredbi iz skupa naredbi SRCa.

ALU naredbe čitaju i mijenjaju sadržaje spremnika. Kako je već opisano upis rezultata ove naredbe izvode na kraju petog koraka, praktički u šestom koraku, dok se operandi dohvaćaju već u drugom koraku. Kako bi se izbjegla pogreška, a ukoliko ne postoji posebno sklopovsko rješenje, potrebno je dvije ALU naredbe odvojiti barem sa četiri nezavisne naredbe. Ali kako je praktički rezultat operacije dostupan već pri kraju trećeg koraka, a i operandi za ALU operaciju potrebni su na početku trećeg koraka, moguće je posebnim sklopovskim rješenjem potreban podatak proslijediti naredbi koja ga treba, prije koraka njegovog upisa u odredišni spremnik. Ovo rješenje već je spomenuto i naziva se prosljeđivanje podatka.

Implementacija prosljeđivanja podatka zahtijeva dodatnu analizu ALU naredbi. Prvo, uz naredbe potrebno je vezati i sljedeća dva podatka: korak kada je rezultat normalno dostupan i korak kada je rezultat najprije dostupan. Kod ALU naredbi informacija (rezultat normalno dostupan)/(rezultat najranije dostupan) označava se kao 6/4. Sljedeći potreban podatak vezan je uz operande, odnosno korak kada je operand normalno potreban i korak kada je operand najkasnije potreban. Kod ALU operacija operandima se pristupa u drugom koraku ali se koriste u trećem koraku. Tako se informacija vezana uz operande (operand normalno potreban)/(operand najkasnije potreban) iznosi 2/3. Temeljen ovih podataka moguće je izračunati minimalni poterbni razmak između dviju ovisnih ALU operacije kao (6-2)/(4-3) = 4/1, odnosno između dvije zavisne ALU naredbe potrebno je ubaciti četiri slobodna mjesta, a korištenje posebnog sklopovlja za prosljeđivanje podatka broj potrebnih slobodnih mjesta reducira se na jedno.

Tablica 5.2. opisuje navedene podatke za sve tipove naredbi, te daje potreban broj slobodnih mjesta između dviju ovisnih naredbi za slučaj bez i sa sklopovljem za prosljeđivanje podataka. Za napomenuti je posebnost naredbe za upis u memoriju, store, koja mora čitati adresu, spremnik rb, i podatak, spremnik ra.

| | Tip naredbi | Upis u spremnike | | | | |
|---|---|---|---|---|---|---|
| | | Podatak dostupan normalno/najprije | | | | |
| | | | alu | load | ladr | brl |
| | | | 6/4 | 6/5 | 6/4 | 6/2 |
| Čitanje iz spremnika normalno/najkasnije | alu | 2/3 | 4/1 | 4/2 | 4/1 | 4/1 |
| | load | 2/3 | 4/1 | 4/2 | 4/1 | 4/1 |
| | ladr | 2/3 | 4/1 | 4/2 | 4/1 | 4/1 |
| | store (rb) | 2/3 | 4/1 | 4/2 | 4/1 | 4/1 |
| | store (ra) | 2/4 | 4/1 | 4/1 | 4/1 | 4/1 |
| | branch | 2/2 | 4/2 | 4/3 | 4/2 | 4/1 |

Tablica 5.2. Podatkovna međuovisnost naredbi koje mijenjaju podatke

Tablica pokazuje kako je sve ovisne naredbe potrebno razdvojiti sa četiri nezavisne naredbe, ali primjenom sklopovlja za prosljeđivanje broj potrebnih slobodnih mjesta značajno se smanjuje.

Provedenim razmatranjima obuhvaćen je samo pristup spremnicima, dok je zanemaren pristup podacima u memoriji. Ovaj slučaj je znatno jednostavniji jer problem može samo postojati ukoliko iza naredbe za upis slijedi naredba za čitanje iz iste lokacije. Ali kako se svaki pristup memoriji odvija u četvrtom koraku ne postoji opasnost pristupa podacima.

**Opasnost u cjevovodu kod naredbi za grananje**. Kod grananja postoji problem da se obavezno izvodi naredba koja slijedi naredbu za grananje neovisno o rezultatu grananja. To znači da je potrebno ubaciti jedno slobodno mjesto iza naredbe za grananje. Postoje neke tehnike predviđanja rezultata grananja ali one prelaze okvire ovog kolegija.

**Detekcija i otklanjanje opasnosti pomoću programa prevodioca**. Program prevodilac može analizom niza naredbi detektirati opasnost dohvata krivih podatak ili izvođenja pogrešne naredbe te preraspodjelom koda ili ubacivanjem slobodnog mjesta razriješiti navedeni problem.

Ovakav pristup ima brojne nedostatke. Prvenstveno značajno povećava zahtjeve na programera koji piše program prevodilac. Program prevodilac je znatno složeniji, veći, skuplji i vjerojatnost pogrešaka je veća. Posebno je složeno ukoliko se provodi naknadna optimizacija koda. Nadalje, program prevodilac izvodi statičku analizu koda. Tako kod uvjetnih grananja mora se analizirati obje situacije, kad se grananje izvodi i kada ne, te generirati ispravan kod za oba slučaja, uzimajući u obzir i ovisnost naredbi koje se izvode prije grananja. Rezultat je uzimanje najgoreg slučaja i generiranja koda za njega, što svakako nije optimalno rješenje.

**Sklopovska detekcija i otklanjanje opasnosti**. Kao posljedica spomenutih nedostatak rješavanja problema detekcije i otklanjanja opasnosti cjevovoda pomoću programa prevodioca, pristupilo se sklopovskom rješavanju spomenutog problema. Temelj sklopovske detekcije je ispitivanje opasnosti u svim koracima u kojima se mogu oni pojaviti, kao što je napravljeno u tablici 5.2. Pristup ovakvom rješavanju problema ilustrirati će se pomoću dvije ALU naredbe. Prvo će se objasniti umetanje praznih mjesta u cjevovod ili nop. Sljedeće činjenice vezane uz cjevovod opisuju situaciju:

1. Minimalni razmak između dvije ovisne naredbe bez sklopovlja za prosljeđivanje podataka iznosi četiri mjesta.
2. Ovisna naredba mora se zaustaviti u drugom koraku, dok se problematična situacija ne razriješi. Razlog je što naredba ne može dohvatiti potreban operand, dok jedna od sljedećih naredbi ne upiše rezultat u odredišni spremnik. Naredba koja je u prvom koraku također se mora zaustaviti.
3. Dvije zavisne naredbe mogu biti razmaknute za jedno, dva ili tri mjesta. Sklopovlje mora detektirati sva tri slučaja.
4. Sve naredbe koje su ušle u cjevovod prije naredbe koja čeka operand moraju se nesmetano izvesti do kraja.

Ako su naredbe u različitim stadijima izvođenja, opasnost mora se detektirati već u trećem koraku. Detekcija opasnosti, zaustavljanje naredbi u cjevovodu te umetanje praznih mjesta, nop, opisati će se RTNom. Upravljačkim signalima opisanim tablicom 5.1. dodati će se oznaka koja se koristi za signalizaciju u kojem koraku je sadržaj spremnika naredbe potreban da bi se otkrila opasnost.

Sljedeći RTN izraz opisuje detekciju opasnosti između koraka dva i tri, zaustavljanje naredbi u prvom i drugom koraku za jedan takt ciklus te umetanje praznog mjesta u treći korak, nop:

alu3 $\wedge$ alu2 $\wedge$ ((ra3 = rb2) $\vee$ ((ra3 = rc2) $\wedge$ $\neg$imm2))
$\rightarrow$ (pauza2: pauza1: op3 $\leftarrow$ 0):

Izraz prikazuje da ako su ALU naredbe u koracima dva i tri i ako je ra u koraku tri jednak rb ili rc u koraku dva (osim ako se radi o neposrednom operandu u kojem slučaju rc nije specificiran), postoji opasnost između naredbi u koracima dva i tri. U tom slučaju generira se signal kojim se zaustavlja izvođenje naredbi u koracima dva i jedan te se umeće prazno mjesto između naredbi u trećem i drugom koraku,. op3 $\leftarrow$ 0

Upravljački signali pauza1 i pauza2 koriste se za zaustavljanje naredbi iz koraka jedan i dva, dok op3 $\leftarrow$ 0 upisuje operacijski kod nop u spremnik naredbe trećeg koraka IR3. Ove aktivnosti realiziraju se na padajući brid takt impulsa. Time se u sljedećem taktu naredba iz trećeg prebacuje u četvrti korak, u trećem koraku je prazno mjesto, a u drugom i prvom koraku ostaju iste naredbe. Slijedi otkrivanje ovisnosti između drugog i četvrtog koraka.

Sklopovlje za otkrivanje ovisnosti naredbi u drugom i četvrtom koraku mora ispitati sljedeća dva slučaja:

1. Opasnost je posljedica već otkrivene ovisnosti među naredbama u drugom i trećem koraku. Već je umetnuto jedno prazno mjesto između naredbi.
2. Ovisnost je otkrivena između naredbi razdvojenih jednom naredbom. Opasnost je otkrivena prvi put kada su te naredbe u drugom i četvrtom koraku.

U oba slučaja opasnost mora se otkriti, zaustaviti se izvođenje naredbi u prvom i drugom koraku te umetnuti prazno mjesto u treći korak. Sljedeći RTN opisuje otkrivanje ovisnosti između naredbi u drugom i četvrtom koraku te generiranje signala za zaustavljanje naredbi u prvom i drugom koraku kao i umetanje praznog mjesta u treći korak:

alu4 $\wedge$ alu2 $\wedge$ ((ra4 = rb2) $\vee$ ((ra4 = rc2) $\wedge$ $\neg$imm2))
$\rightarrow$ (pauza2: pauza1: op3 $\leftarrow$ 0):

Ovaj opis jako je sličan prethodnom uz minimalnu razliku što se uspoređuju spremnici četvrtog koraka sa spremnicima drugog koraka. Ukoliko je već prije umetnuto prazno mjesto ono se prebacuje u četvrti korak, a u treći se ubacuje novo prazno mjesto. Naredbe iz drugog i prvog koraka ostaju zaustavljene.

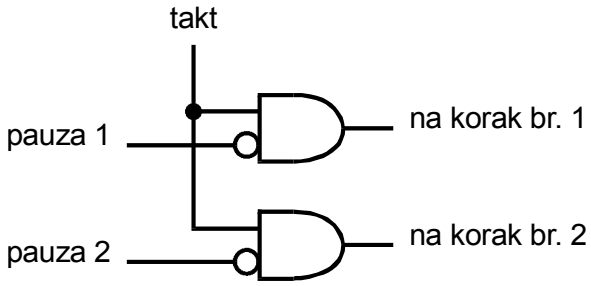
Otkrivanje ovisnosti između naredbi u petom i drugom koraku identična je prethodno opisanom postupku uz razliku koji se spremnici uspoređuju:

alu5 $\wedge$ alu2 $\wedge$ ((ra5 = rb2) $\vee$ ((ra5 = rc2) $\wedge$ $\neg$imm2))
$\rightarrow$ (pauza2: pauza1: op3 $\leftarrow$ 0):

Nakon petog koraka, naredba koja je zaustavila izvođenje u cjevovodu, završila je s izvođenjem i upisala rezultat u odredišni spremnik. Više ne postoji opasnost u cjevovodu te može nastaviti s izvođenjem naredbe iz drugog koraka. U najgorem slučaju su dvije ovisne naredbe jedna iza druge pa je potrebno umetnuti tri prazna mjesta.

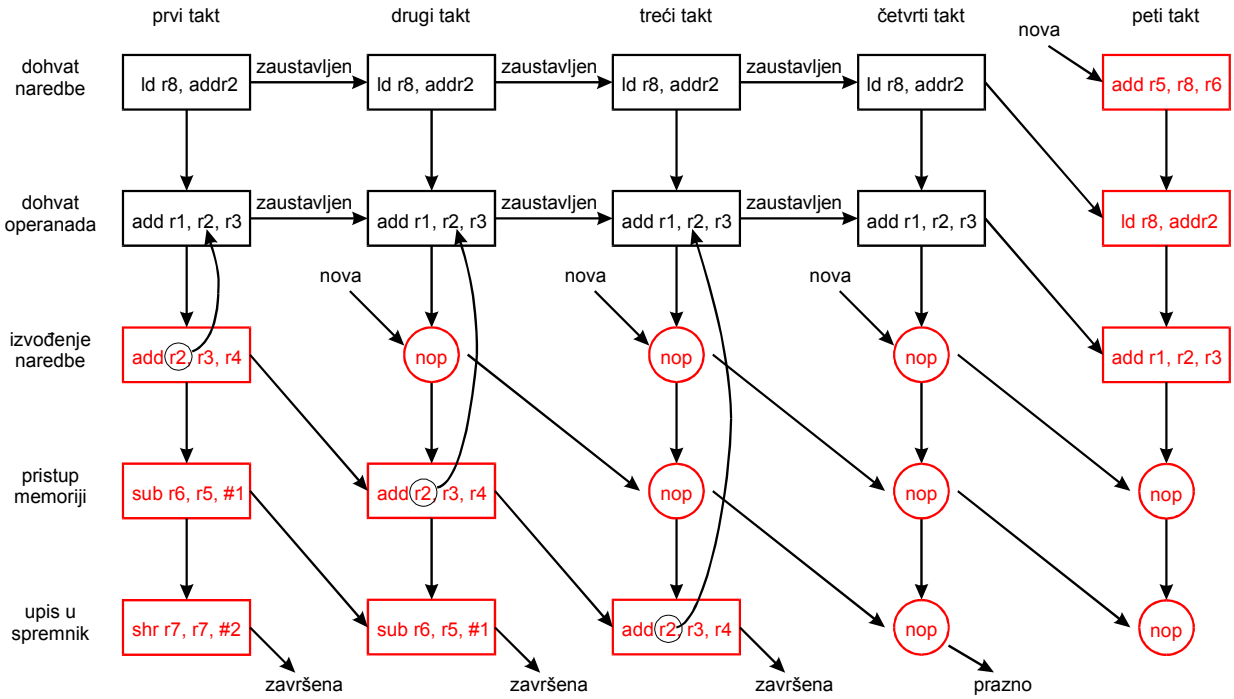
Sličan mehanizam za otkrivanje opasnosti unutar cjevovoda mora se projektirati za sve moguće međuovisnosti naredbi.

Slika 5.11 prikazuje moguće sklopovsko rješenje generiranja signala za zaustavljanje naredbi u prvom i drugom koraku cjevovoda. Realizacija se temelji na jednostavnom odvajanju takt signala iz tih koraka.




Slika 5.11. Sklopovsko rješenje zaustavljanja naredbi u prvom i drugom koraku cjevovoda.

Na slici 5.12. prikazan je primjer zaustavljanja naredbi i umetanje praznih mjesta u cjevovodu koje je posljedica međuovisnosti naredbi. U prvom takt impulsu otkrivena je međuovisnost naredbe u trećem koraku, koja mijenja sadržaj spremnika r2, i naredbe drugom koraku koja koristi sadržaj ovog spremnika za ALU operaciju. U sljedeća tri takta umeću se prazna mjesta između ovih naredbi, a naredbe u prvom i drugom koraku cjevovoda su zaustavljene sve dok se u petom koraku ne razriješi opasnost.




Slika 5.12. Primjer međuovisnosti dviju naredbi , umetanja praznih mjesta i zaustavljanja naredbi iz prvog i drugog koraka cjevovoda.

**Otkrivanje opasnosti pomoću sklopova, rješenje prosljeđivanja podataka**. Prosljeđivanje podataka donekle rješava problem opasnosti ovisnosti podatka naredbi. Potreban podatak prosljeđuje se naredbi koja ga koristi čim je dostupan i prije nego što se isti upiše u odredišni spremnik. Zavisno o vrsti ovisnosti među naredbama potrebno je možda ubaciti samo jedno prazno mjesto između njih. Temeljni koncept je krajnje jednostavan iako sama sklopovska implementacija je složena. Ovisnost među naredbama potrebno je otkriti, prazno mjesto se umeće samo u slučaju kada potreban podatak još nije dostupan (izračunat) i zavisna naredba se zaustavlja. Prosljeđivanje može rezultirati izvođenjem bez zaustavljanja ukoliko se naredbe prosljeđuju na izvođenje bez ovisnosti između dviju susjednih naredbi.

Projektiranje i realizacija sklopovlja za prosljeđivanje podataka opisati će se ponovo na primjeru SRCa. Prema tablici 5.2. može se zaključiti da dvije susjedne naredbe koje su međuovisne mogu se neometano izvoditi ukoliko se na odgovarajući način rezultat prve naredbe proslijedi kao operand drugoj naredbi. Ovisnost, odnosno opasnost potrebno je ispitivati tek u trećem koraku. Tada se izvodi alu operacija. Operand potreban ovoj naredbi može se nalaziti ili u četvrtom ili u petom koraku. Kada sklopovlje otkrije ovisnost među naredbama, prosljeđuje podatak iz četvrtog ili petog koraka na jedan od ulaza ALU u ovisnosti o kojoj se naredbi radi. Za realizaciju navedenog potrebno je i proširiti spremnik naredbe u trećem koraku, IR3, za polja rb i rc kako bi ove informacije logika za otkrivanje opasnosti mogla otkriti ovisnost među naredbama i na odgovarajući način proslijedila operand. Ovisnost je opisana na sljedeći način:

alu5 $\wedge$ alu3 $\rightarrow$ ((ra5 = rb3) $\rightarrow$ X $\leftarrow$ Z5:
(ra5 = rc3) $\rightarrow$ Y $\leftarrow$ Z5); 3 – 5 ovisnost

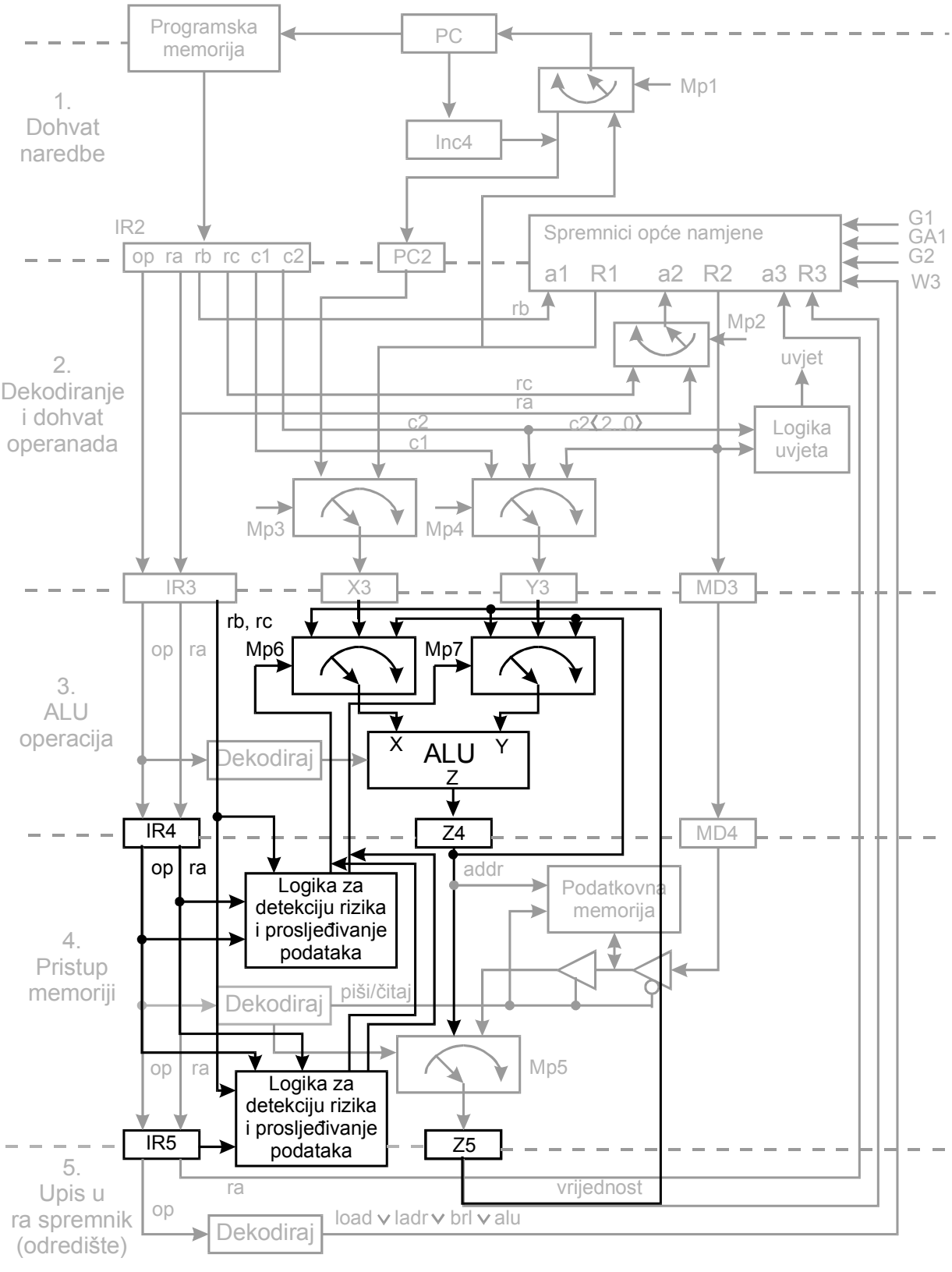
alu4 $\wedge$ alu3 $\rightarrow$ ((ra4 = rb3) $\rightarrow$ X $\leftarrow$ Z4:
(ra4 = rc3) $\rightarrow$ Y $\leftarrow$ Z4); 3 – 4 ovisnost

gdje su X i Y lijevi i desni ulazi ALU. U ovom slučaju nije potrebno umetanje praznih mjesta u cjevovod. Sklopovsko rješenje sustava za otkrivanje opasnosti u cjevovodu i prosljeđivanje podataka prikazana je na slici 5. 15. Sklopovlje uzima informacije iz op, rb i rc polja spremnika IR3 te op i ra polja spremnika IR4 i IR5. Ovim sklopovima otkriva se opasnost u cjevovodu koja je posljedica međuovisnosti dviju ALU naredbi. Ukoliko opasnost je otkrivena između koraka 3 i 5 te koraka 3 i 4, multipleksori Mp6 i Mp7 spajaju odgovarajuće signale (izlaz spremnika i Z4 ili Z5) na ulaze ALU. Važno je napomenuti kako međuovisnost može biti višestruka i obuhvaća istovremeno više koraka. Primjer je sljedeći odsječak koda:

```
add     r1, r3, r2      ; naredba A u petom koraku izvođenja
add     r1, r4, r1      ; naredba B u četvrtom koraku izvođenja
add     r5, r1, r1      ; naredba C u trećem koraku izvođenja
```

Kada su naredbe A i B u trećem i četvrtom koraku izvođenja, rezultat naredbe A se prosljeđuje naredbi B. U sljedećem taktu naredbe A i B prelaze u korake četiri i pet dok naredba C ulazi u treći korak. Naredba C za operande ima odredišni spremnik prethodne naredbe B, ali isti spremnik može se detektirati kao potencijalna opasnost jer je isti odredišni spremnik naredbe A. Ipak logika mora odrediti da je naredbi C potrebno proslijediti operand iz naredbe B, a odbaciti operand naredbe A. Ovo znači da je prioritetnija ovisnost 3 – 4 nad ovisnošću 3 – 5.

Dodatno sklopovlje potrebno je za otkrivanje ovisnosti između preostalih naredbi koje procesor može izvesti.




Slika 5.13. Sklopovlje za otkrivanje opasnosti u cjevovodu i prosljeđivanje podataka.

**Posebna stanja i cjevovod**. Unutarnja i vanjska posebna stanja moraju se također uzeti u obzir prilikom projektiranja strukture cjevovoda. Obrada posebnih stanja kod procesora sa cjevovodom, naročito kod procesora koji imaju i više cjevovoda, tzv. procesora koji istovremeno obrađuju više naredbi ili superskalarni procesori posebno je složena. Kod takvih procesora projektanti su odustali od nakane da projektiraju sustav za obradu posebnih stanja koji je u stanju u potpunosti sačuvati stanje procesora i u potpunosti ga obnoviti nakon obrade posebnog stanja. Posebna stanja kod spomenutih procesora nazivaju se i neprecizna posebna stanja (*imprecise exceptions*).

Ipak većinu posebnih stanja, sustav za njihovu obradu i kod spomenutih procesora uspješno obrađuje. Navedena posebna stanja nazivaju se precizna posebna stanja (*precise exceptions*). Kod preciznih posebnih stanja jednoznačno je određeno što se dešava s naredbama prije za vrijeme i nakon njihove obrade. Vanjska posebna stanja ili prekidi ispituju se prije nego se uzme sljedeća naredba u cjevovod. Ako je došlo do prekida, pohranjuje se sadržaj programskog brojila i zamjenjuje se sadržajem vektora prekida. Naredbe prekinutog programa koje se nalaze u cjevovodu nastavljaju se izvoditi. Nasuprot prekidima, interno izazvana posebna stanja znatno su složenija. Naime, oni se mogu pojaviti u bilo kojoj fazi obrade naredbe. Tako npr. nepostojeća naredba dekodira se u drugom koraku, alu pogreška u trećem, pogreške prilikom pristupa memoriji u četvrtom koraku cjevovoda. Pojavom navedenih situacija postavlja se pitanje što činiti s naredbama koje su ušle u cjevovod iza naredbe koja je izazvala posebno stanje. Idealno rješenje je sačuvati sadržaje spremnika faze u kojoj je došlo do posebnog stanja za naknadnu analizu pomoću procedure za obradu posebnih stanja, naredbu zamijeniti praznim mjestom, a naredbe koje su ušle u cjevovod nakon ove naredbe poništiti i ponovo izvesti nakon procedure za obradu posebnog stanja.

**Performanse procesora sa cjevovodom**. Kao i u prethodnim slučajevima moguće je provesti kvantitativnu analizu performansi procesora sa cjevovodom. Kao prva pretpostavka neka je frekvencija takta identična onoj kao i kod procesora bez cjevovoda. Ubrzanje se računa prema izrazu:

$$\%\text{ubrzanj} = \frac{(\text{IC} \times \text{CPI} \times \tau)_{\text{bez cjevovoda}} - (\text{IC} \times \text{CPI} \times \tau)_{\text{sa cjevovodom}}}{(\text{IC} \times \text{CPI} \times \tau)_{\text{sa cjevovodom}}} \times 100$$

Kako broj naredbi je jednak za oba slučaja kao i trajanje takta slijedi:

$$\%\text{ubrzanj} = \frac{\text{CPI}_{\text{bez cjevovoda}} - \text{CPI}_{\text{sa cjevovodom}}}{\text{CPI}_{\text{sa cjevovodom}}} \times 100$$

Kod jednosabirničkog SRCa prosječno se izvede jedna naredba u pet taktova dok kod procesora sa cjevovodom idealno jedna naredba u jednom taktu. Ubrzanje tada iznosi:

$$\%\text{ubrzanj} = \frac{\text{CPI}_{\text{bez cjevovoda}} - \text{CPI}_{\text{sa cjevovodom}}}{\text{CPI}_{\text{sa cjevovodom}}} \times 100 = \frac{5-1}{1} \times 100 = 400\%$$

Kod realnije pretpostavke da se svakih pet naredbi ubacuje prosječno jedno prazno mjesto, odnosno izvede se prosječno četiri naredbe u pet taktova dobiva se:

$$\%\text{ubrzanj} = \frac{\text{CPI}_{\text{bez cjevovoda}} - \text{CPI}_{\text{sa cjevovodom}}}{\text{CPI}_{\text{sa cjevovodom}}} \times 100 = \frac{5-5/4}{5/4} \times 100 = 375\%$$

što je također veliko ubrzanje.