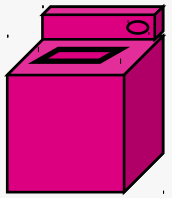
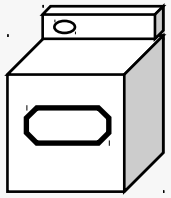
Arhitektura računara

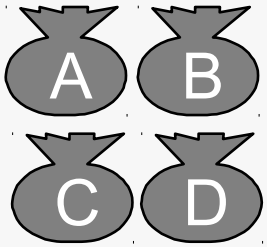
dr.sc. Amer Hasanović

Praonica odjeće

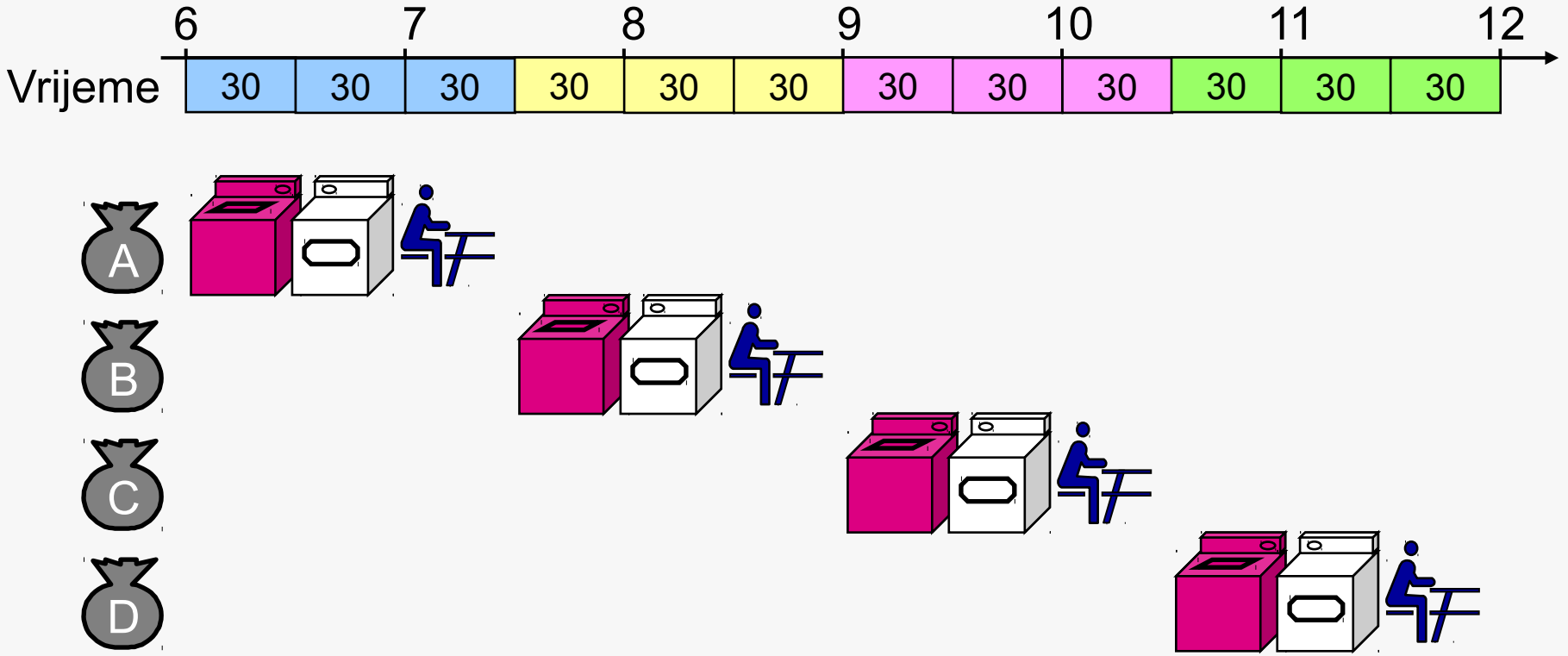
- Praonica operira u tri faze:
- Pranje u mašini:
- Sušenje u sušilici:
- Peglanje i slaganje odjeće
  - Neka svaka faza zahtijeva 30 min.
  - predpostavimo da četiri klijenta dolaze u isto vrijeme





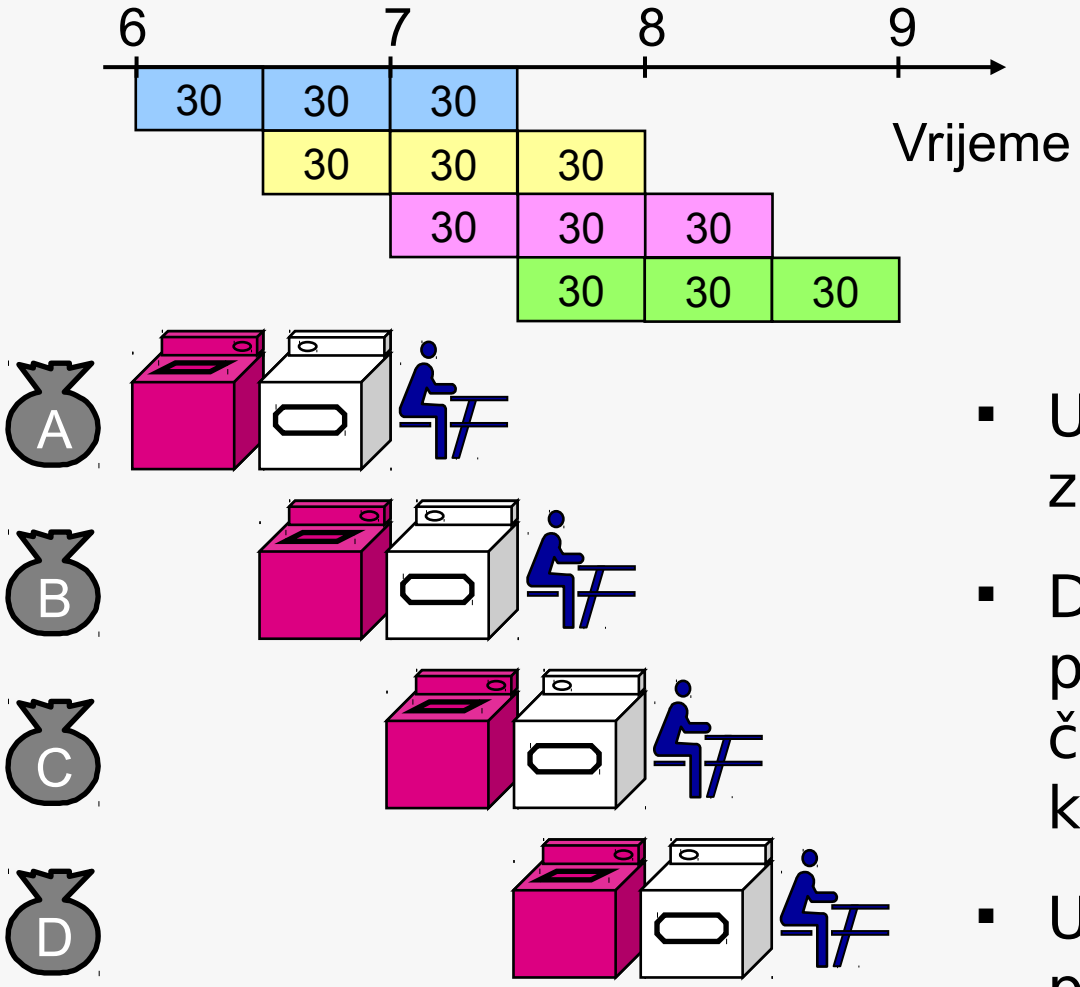


Sekvencijalna praonica



- Ukupno potrebno 6 sati za 4 klijenta

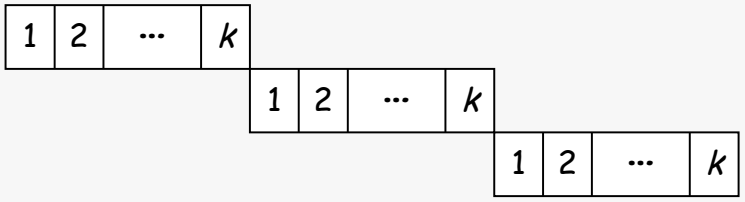
Praonica sa cjevovod implementacijom



- Ukupno potrebno 3 sata za 4 klijenta
- Dvostruko ubrzanje prilikom procesiranja četiri istovremena klijenta
- Ukupno vrijeme procesiranja jednog klijenta još uvijek je 1 sat i 30 min

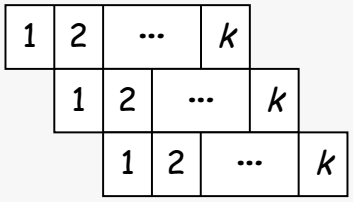
Generalizacija

- Neka konstantno pristižu zadaci od kojih se svaki može podjeliti u *k* podzadataka
  - Svaki podzadatak izvodi se u separatnoj fazi koja zahtijeva jednu vremensku jedinicu
  - Ukupno vrijeme izvršenja jednog zadatka zahtijeva *k* vremenskih jedinica
- Cjevovod implementacija započinje drugi zadatak prije kraja prvog zadatka
- U datom trenutku maksimalno *k* faza radit će paralelno na k različitih zadataka



Bez cjevovoda

jedan zadatak završava se za *k* vremenskih jedinica



Sa cjevovodom

jedan zadatak završava se u svakoj vremenskoj jednici

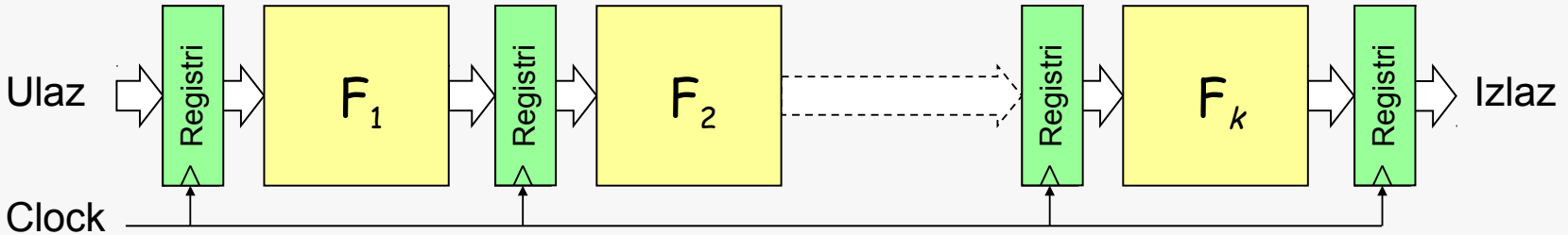
Performanse

- Neka je t_i vrijeme procesiranja faze F_i
- Ciklus clock signala je tada $t = \max(t_i)$ tj frekvencija clock signala je $f = 1/t = 1/\max(t_i)$
- Cjevovod procesira n zadataka u $k+n-1$ ciklusa
  - k ciklusa potrebno da cjevovod procesira prvi zadatak
  - $n-1$ ciklus potrebno da se procesira preostalih $n-1$ zadataka
- Ubrzanje u odnosu na serijski slučaj je:

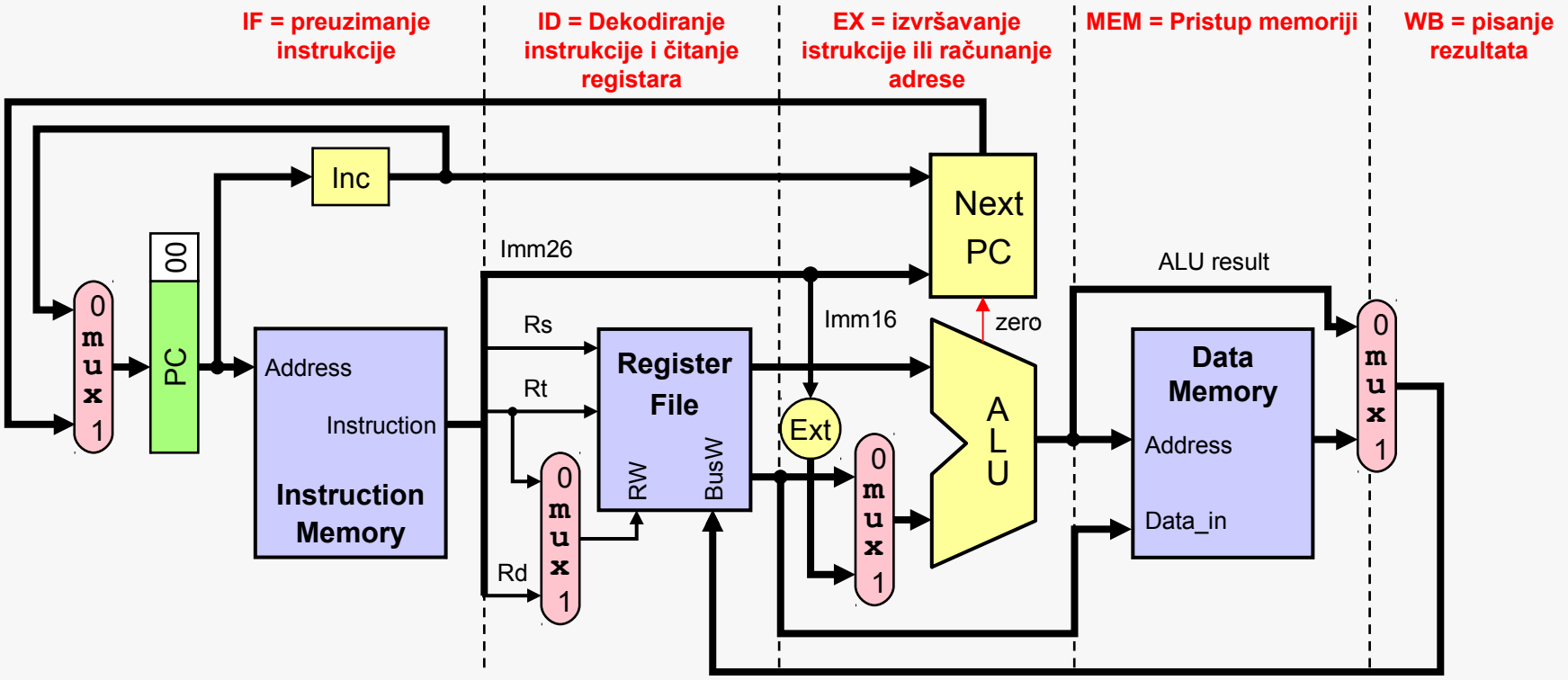
$$U = \frac{\text{br. ciklusa serija}}{\text{br. ciklusa cjevovod}} = \frac{nk}{k+n-1} \qquad U \to k \ \text{ za veliko } n$$

Sinhroni cjevovod

- Koristi registre za simultano snimanje rezultata između faza
- Na početku svakog novog ciklusa clock signala u registrima se nalaze rezultati iz prethodnih faza
- Svaka faza cjevovoda implementira se kao kombinatorno kolo
- Poželjno je izbalansirati trajanje svake faze
  - Clock ciklus određuje se na osnovu trajanja najduže faze

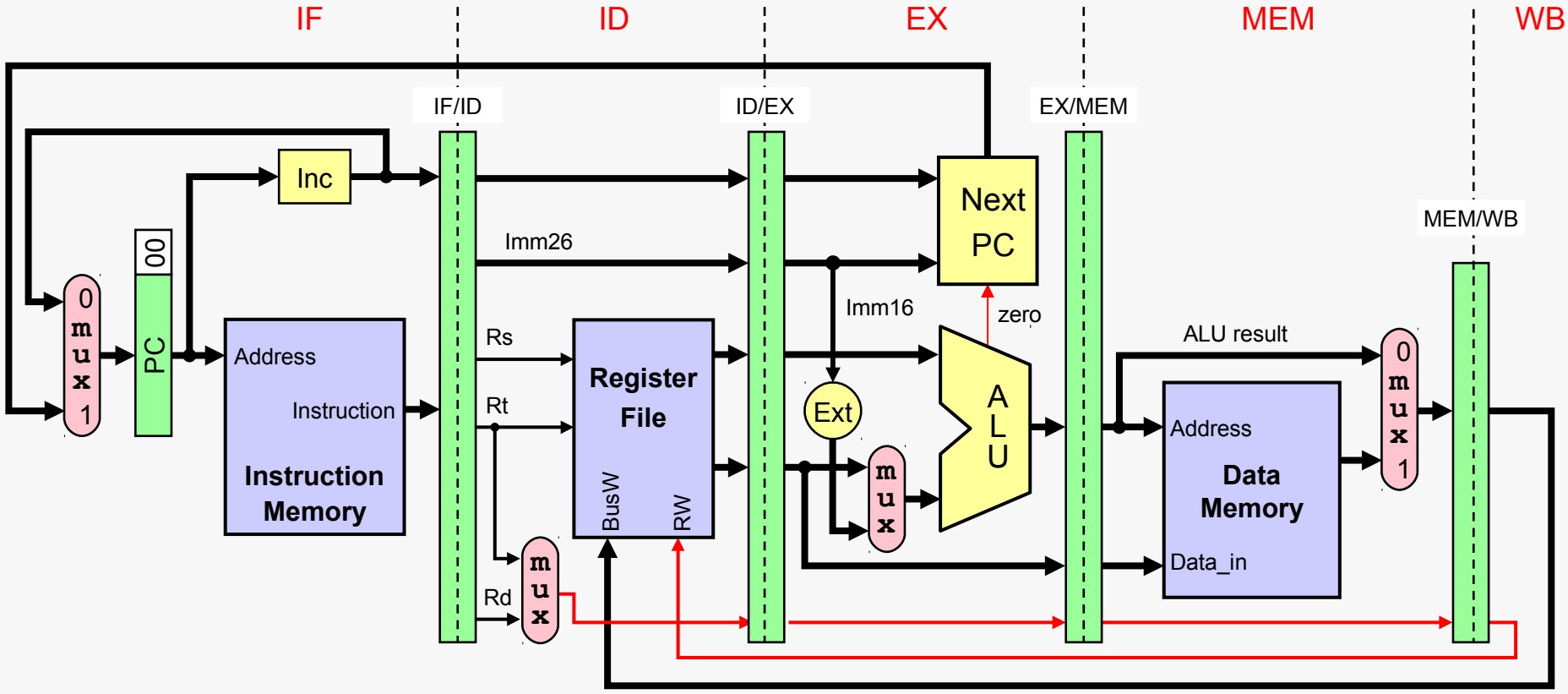


Jednociklusni datapath



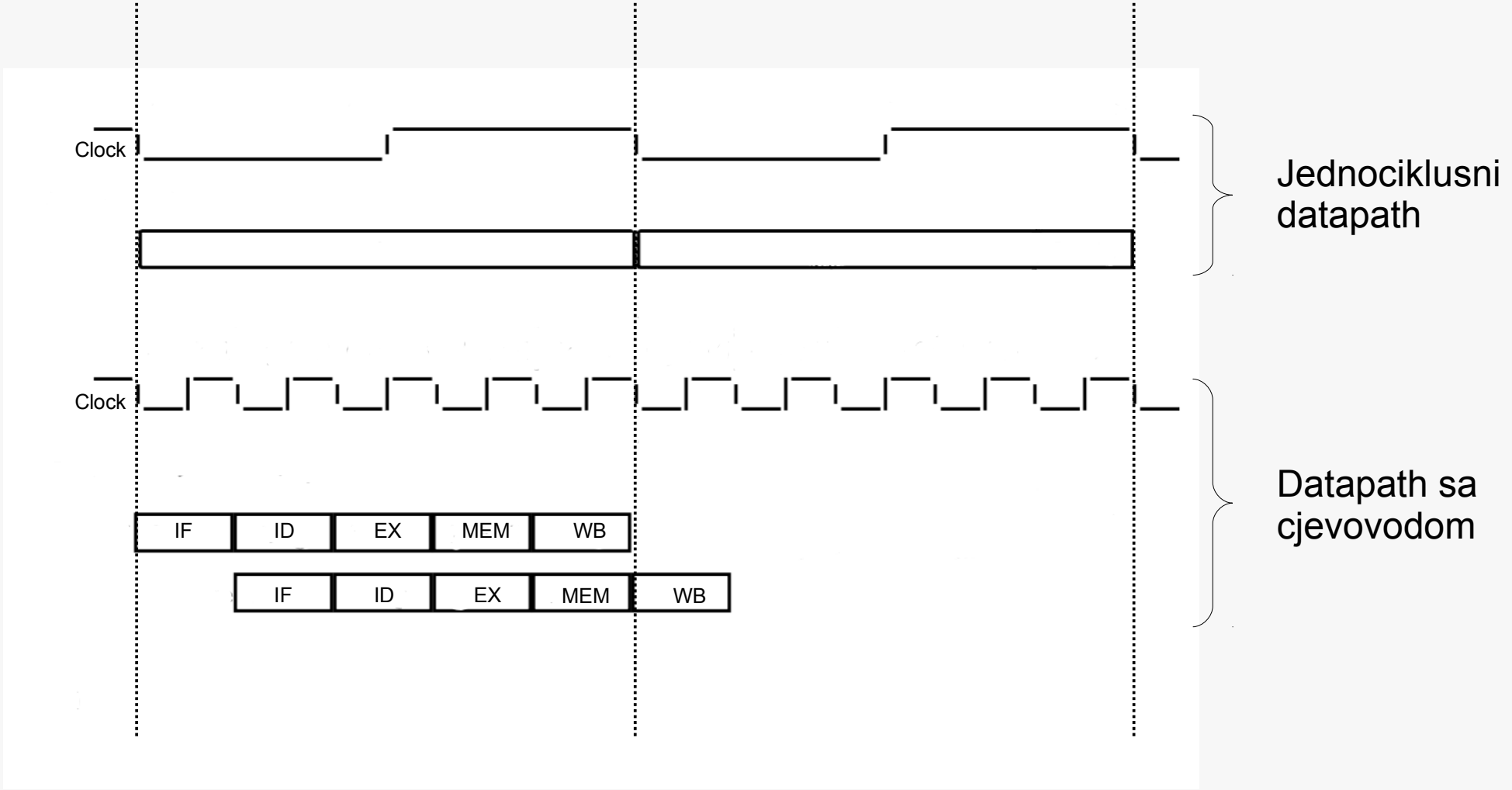
Datapath sa cjevovodom

- Između faza dodati cjevovod registri
  - Cjevovod registri se označavaju na osnovu faza koje razdvajaju



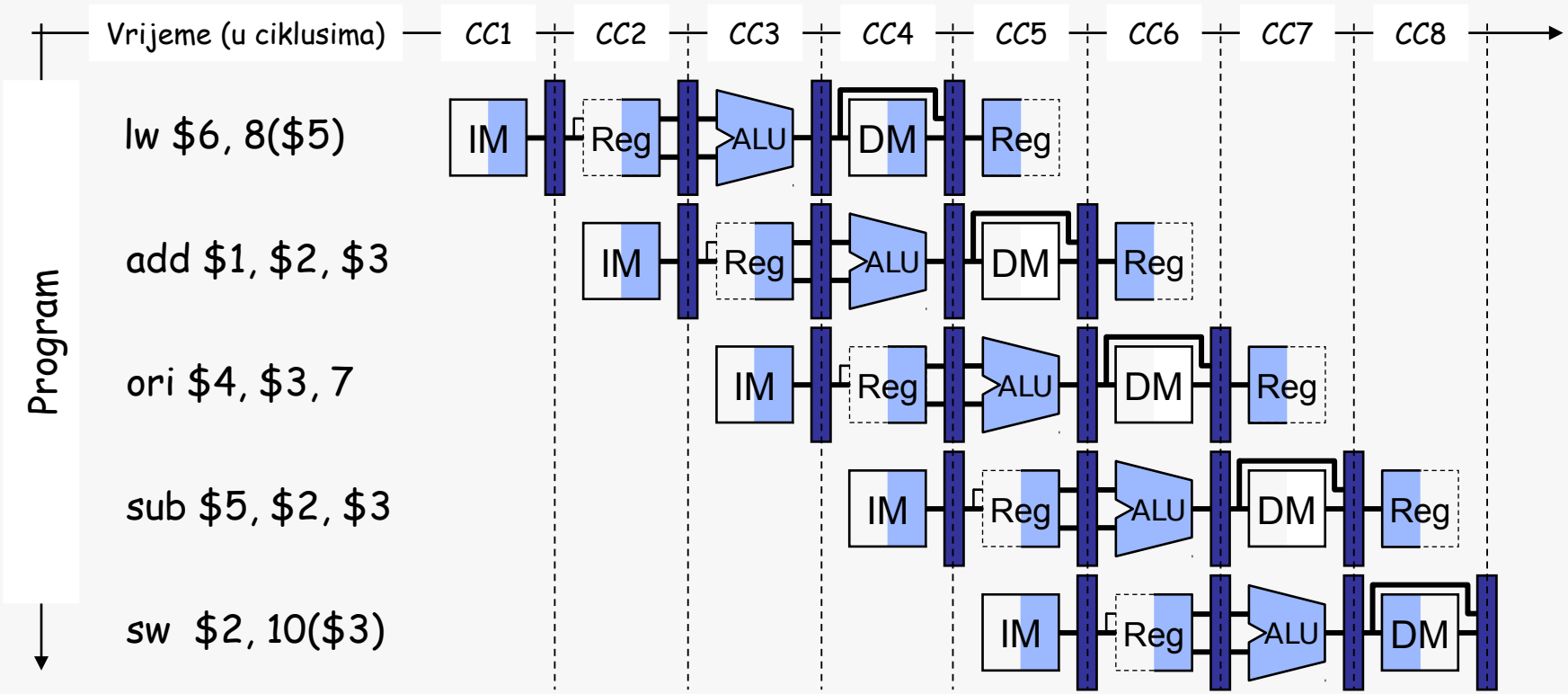
Vremenski dijagram

- Dijagram izvršenja dvije sukcesivne instrukcije

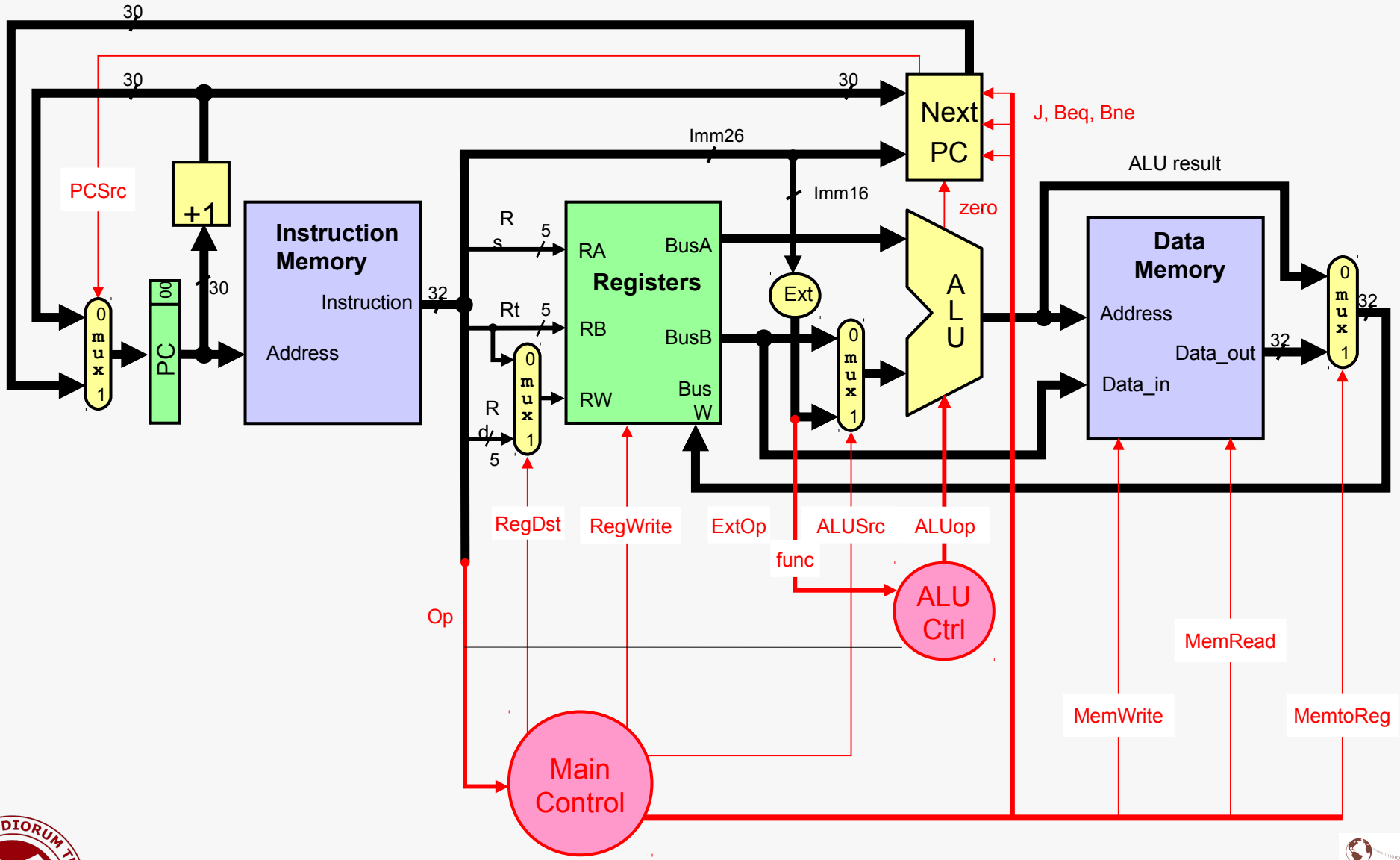


Grafički prikaz izvršenja u cjevovodu

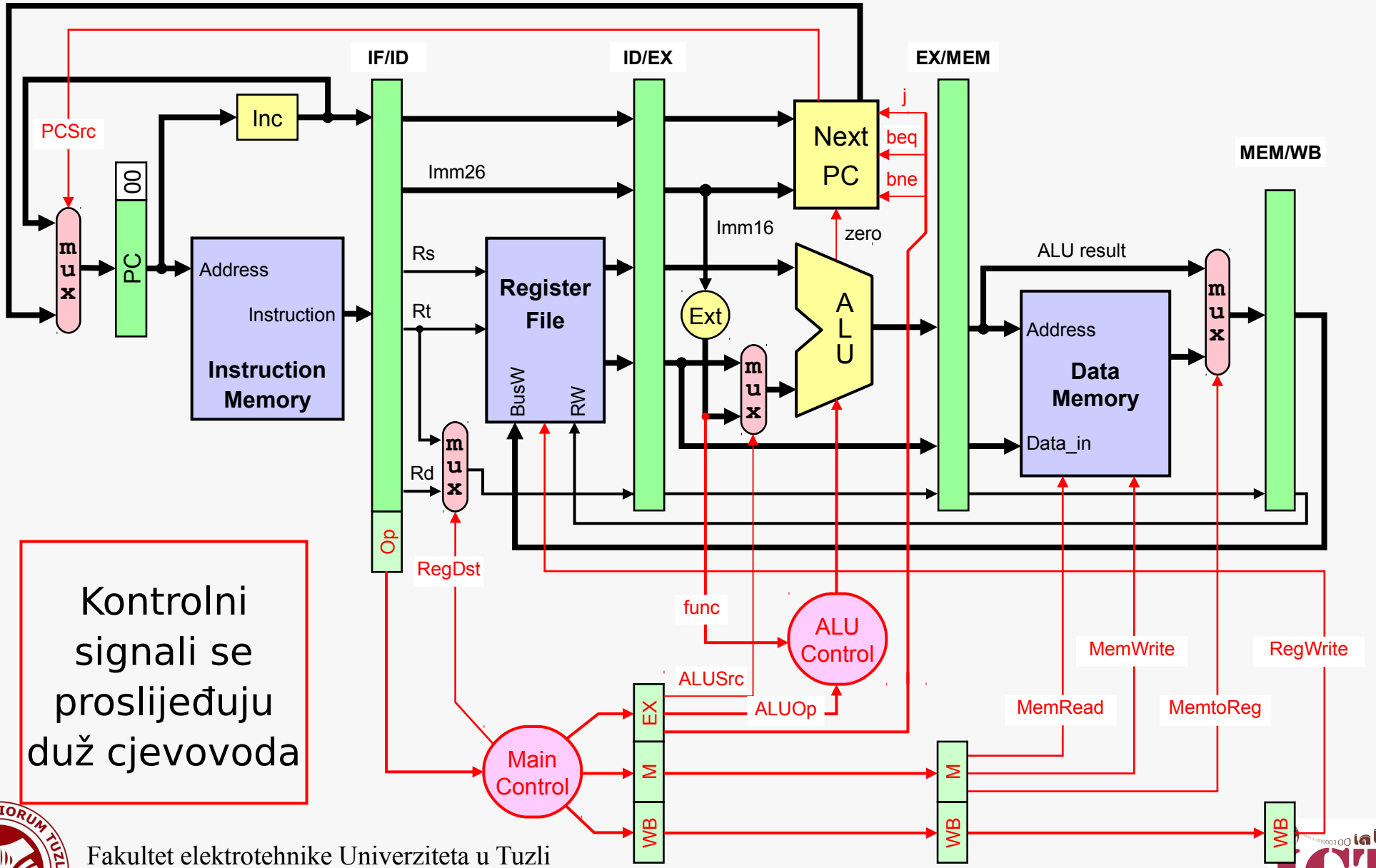
- Tokom više ciklusa clock signala paralelno se izvodi više instrukcija
- Grafik pokazuje korištenje pojedinih resursa u cjevovodu



Kontrola jednociklusnog CPU-a



Kontrola CPU-a sa cjevovodom



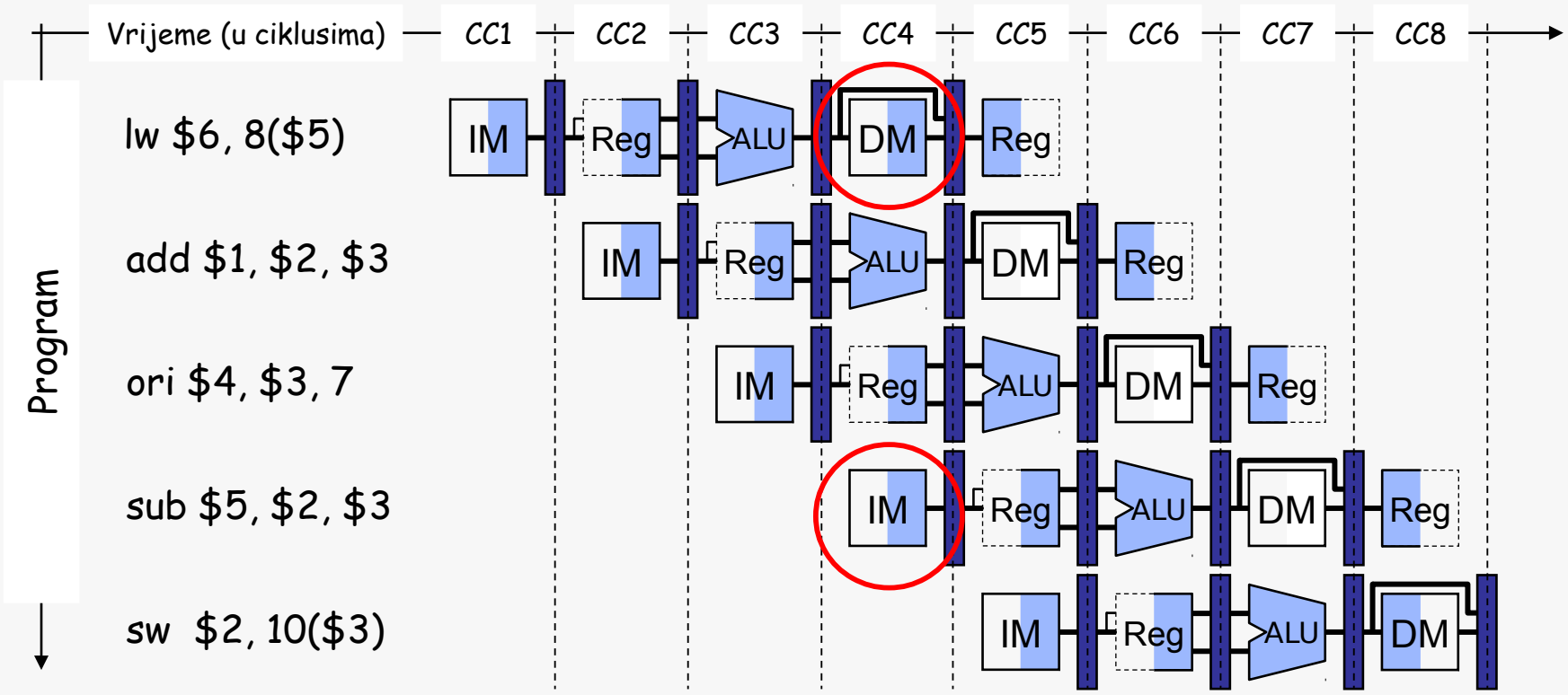
Kontrolni signali se proslijeđuju duž cjevovoda

Cjevovod hazardi

- Situacije koje nastaju u cjevovodu koje bi izazvale pogrešno izvršenje instrukcije u slučaju da se neka faza cjevovoda izvrši u predviđenom ciklusu:
  - Strukturni hazard (structural hazard)
    - Kada dvije instrukcije trebaju istovremeno da koriste istu funkcionalnu jedinicu
  - Podatkovni hazard (data hazard)
    - Kada izvršenje instrukcije ovisi od rezultata instrukcije koja se još uvijek izvršava u cjevovodu
  - Kontrolni hazard (control hazard)
    - Kada preuzimanje (fetch) instrukcije ovisi o rezultatu izvršenja instrukcije koja je u cjevovodu
- Hazardi komplikuju implementaciju kontrole i degradiraju performanse

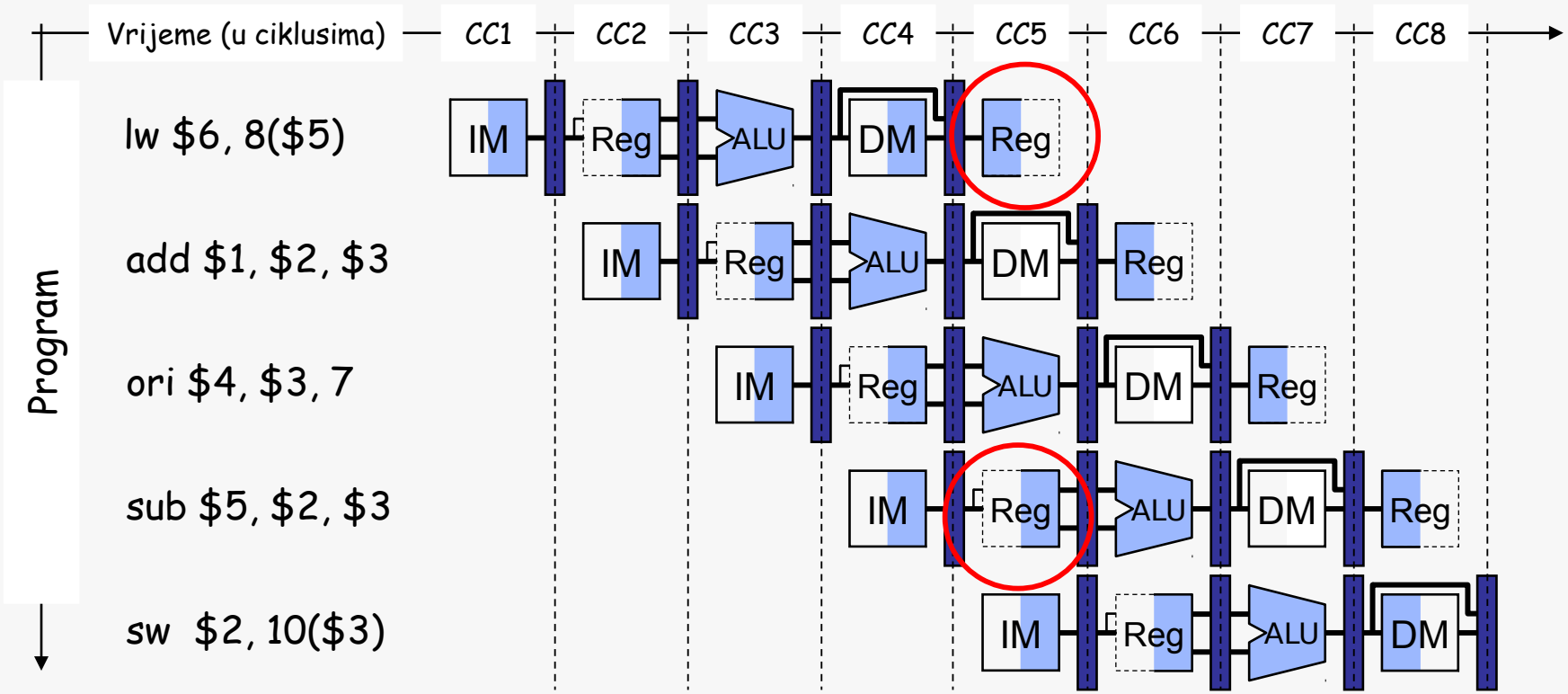
Strukturni hazard 1

- Za slučaj da postoji samo jedna memorija:
  - Skupo postojanje dvije memorije
  - Riješava se uvođenjem L1 keš memorija za podatke i instrukcije



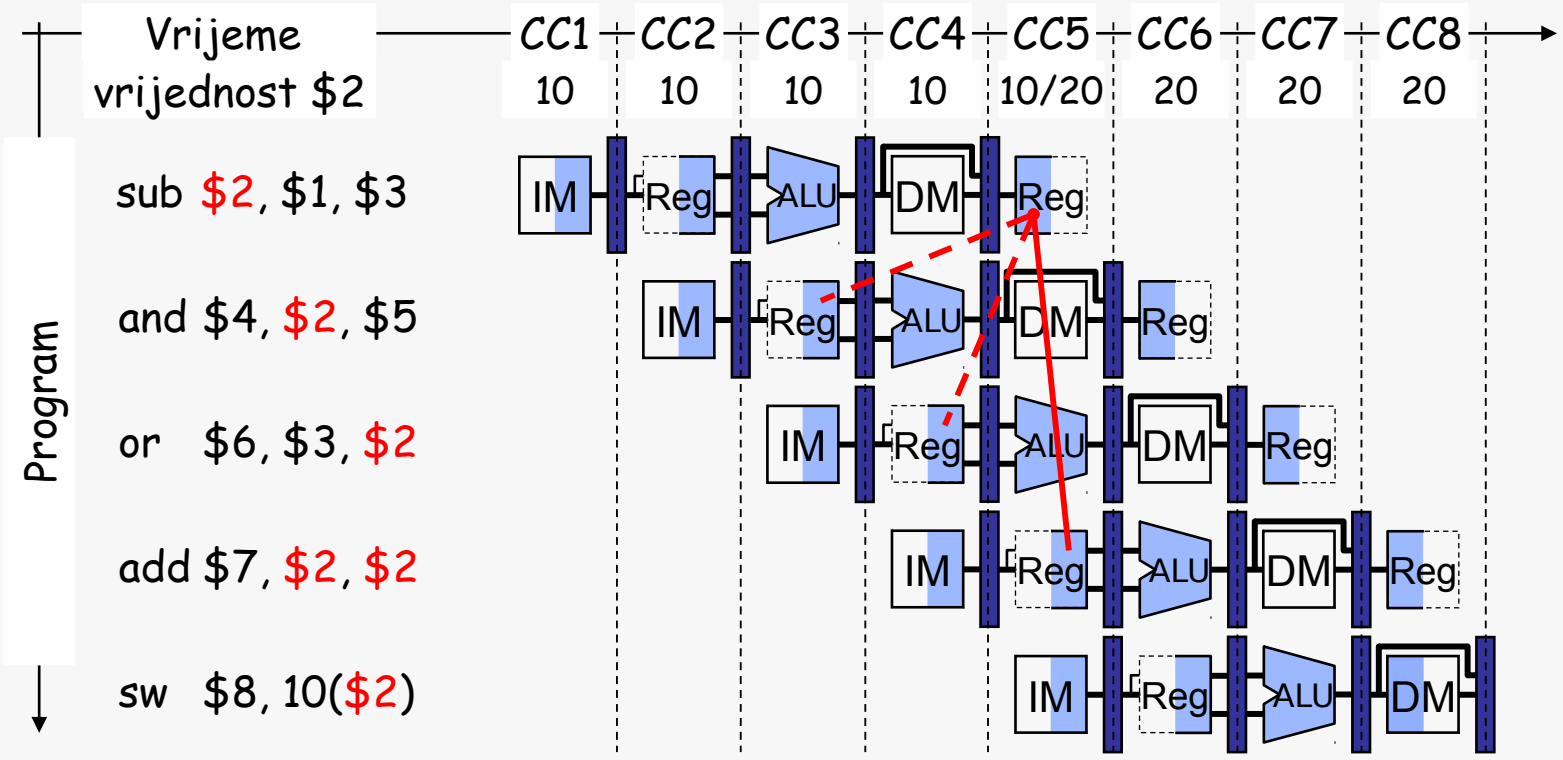
Strukturni hazard 2

- Istovremeno čitanje i pisanje registara:
  - Pristup registrima veoma brz
  - Riješenje se nalazi na način da se pisanje obavlja u prvoj polovini ciklusa a čitanje u drugoj polovini ciklusa

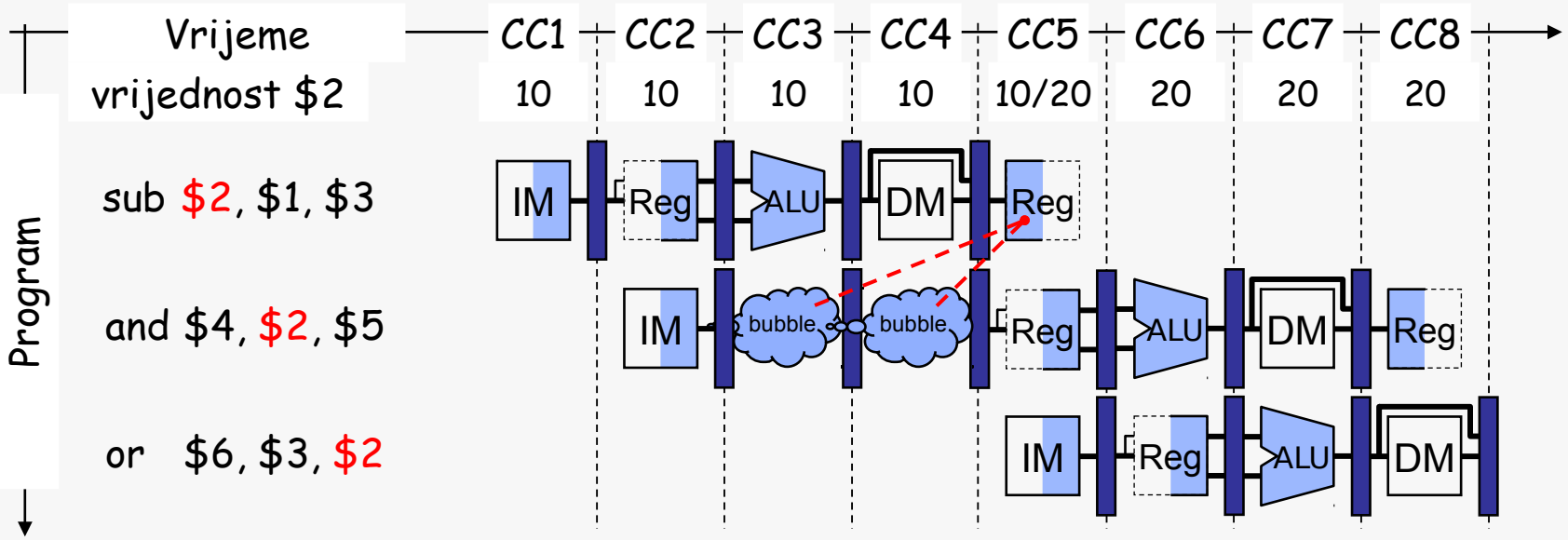


Podatkovni hazard

- Međuovisnost dvije instrukcije koje se pojavljuju blizu u kodu dovodi do podatkovnog hazarda
- Čitanje neposredno nakon pisanja (RAW)
  - I instrukcija dolazi prije od J instrukcije a I zapisuje rezultat u registar koji potražuje instrukcija J npr:



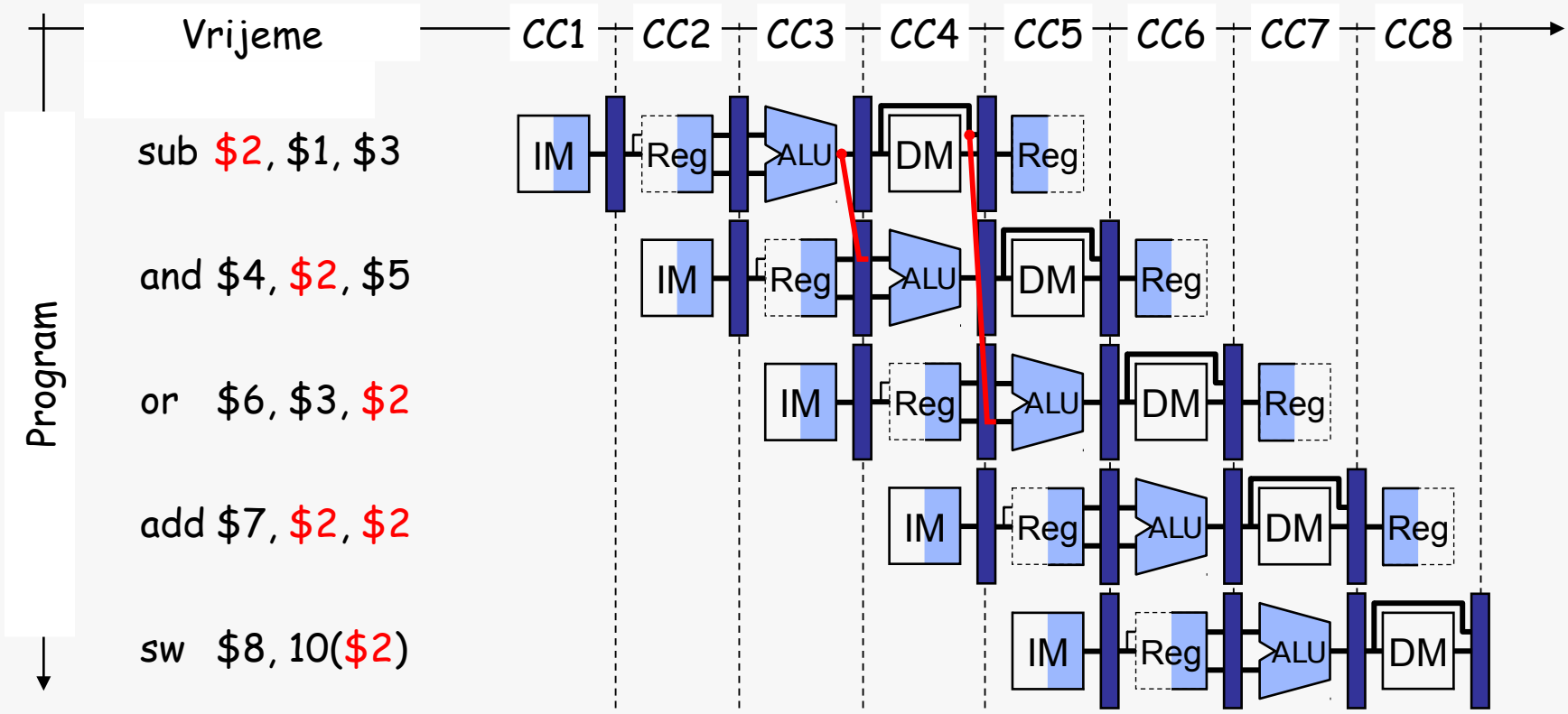
Rješenje 1: Odgađanje izvođenja



- Odgoditi dekodiranje instrukcije koja čita, dok prethodna faza ne završi upisivanje u registar (tzv pipeline stall)
- Bubble je NOP operacija koja ne modificira ni registre ni memoriju
  - troši clock cikluse ne radeći ništa

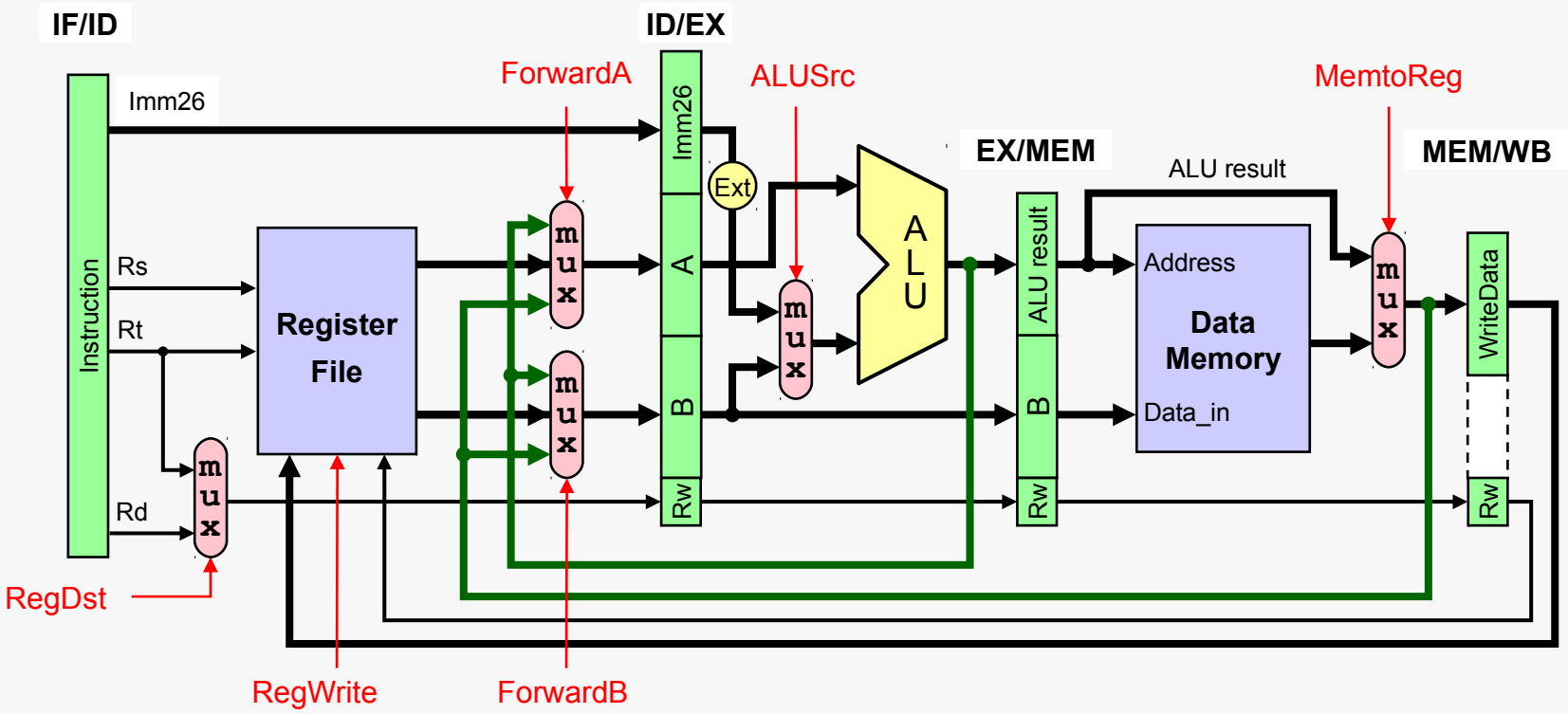
Rješenje 2: Proslijeđivanje (forwarding)

- Rezultat iz ALU jedinice prosljeđuje se slijedećoj instrukciji čim postane na raspolaganju



Implementacija proslijeđivanja

Dvije MUX jedince skupa sa njihovim kontrolnim signalima ForwardA and ForwardB dodaju se u datapath



RAW hazard detekcija

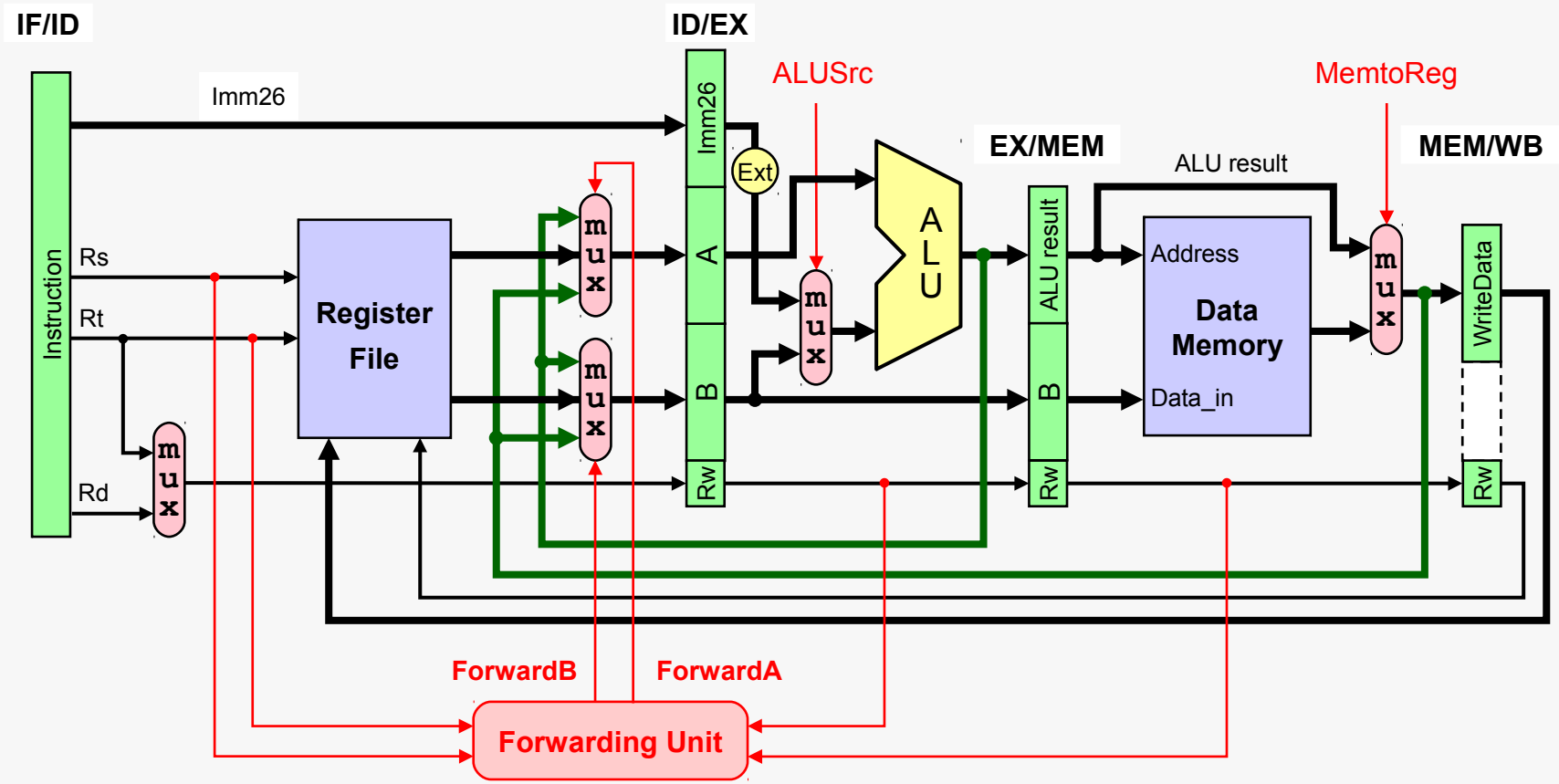
- Trenutna instrukcija koja se dekodira je u `IF/ID` registru
- Posljednja instrukcija je u `ID/EX` registru
- Pretposljednja instrukcija je u `EX/MEM` registru
- RAW hazard uslovi:
  - `IF/ID.Rs=ID/EX.Rw`
  - `IF/ID.Rt=ID/EX.Rw`

  } Raw hazard detektvan sa posljednjom instrukcijom

  - `IF/ID.Rs=EX/MEM.Rw`
  - `IF/ID.Rt=EX/MEM.Rw`

  } Raw hazard detektovan sa pretposljednjom instrukcijom

Kontrola proslijeđivanja



Primjer

Instrukcije:

```
lw    $4, 100($9)
add   $7, $5, $6
sub   $8, $4, $7
```